Optimisation of Piping Network Design for District Cooling System

by

Lok Shun Apple CHAN

A thesis submitted in partial fulfillment

of the award of

Doctor of Philosophy

of

De Montfort University

June 2008

ABSTRACT

A district cooling system (DCS) is a scheme for centralised cooling energy distribution which takes advantage of economies of scale and load diversity. A cooling medium (chilled water) is generated at a central refrigeration plant and then supplied to a district area, comprising multiple buildings, through a closed-loop piping circuit. Because of the substantial capital investment involved, an optimal design of the distribution piping configuration is one of the crucial factors for successful implementation of a district cooling scheme. Since there exists an enormous number of different combinations of the piping configuration, it is not feasible to evaluate each individual case using an exhaustive approach.

This thesis examines the problem of determining an optimal distribution piping configuration using a genetic algorithm (GA). In order to estimate the spatial and temporal distribution of cooling loads, the climatic conditions of Hong Kong were investigated and a weather database in the form of a typical meteorological year (TMY) was developed. Detailed thermal modelling of a number of prototypical buildings was carried out to determine benchmark cooling loads.

A novel Local Search/Looped Local Search algorithm was developed for finding optimal/near-optimal distribution piping configurations. By means of computational experiments, it was demonstrated that there is a promising improvement to GA performance by including the Local Search/Looped Local Search algorithm, in terms of both solution quality and computational efficiency. The effects on the search

performance of a number of parameters were systematically investigated to establish the most effective settings.

In order to illustrate the effectiveness of the Local Search/Looped Local Search algorithm, a benchmark problem – the optimal communication spanning tree (OCST) was used for comparison. The results showed that the Looped Local Search method developed in this work was an effective tool for optimal network design of the distribution piping system in DCS, as well as for optimising the OCST problem.

ACKNOWLEDGEMENT

I would like to thank Professor V.I. Hanby, my supervisor at the De Montfort University, who has given me invaluable guidance and support throughout this research. His precious visions and precise advices have been the key factors leading to the success of this study.

I also thank my local supervisor Dr T.T. Chow of the City University of Hong Kong who has given me great motivation for my research work. Thanks also go to my second supervisor Dr Ljiljana Marjanovic-Halburd at the De Montfort University.

Sincere thanks are expressed to Dr Square K.F. Fong, one of my best colleagues, for his encouragement and sharing of his valuable experience. I will never forget the friendly research atmosphere coming from many kind and interesting people, in particular, Ms. Maggie Y.Y. Chang, Ms. Zoe S.Y. Chang, Ms. C.L. Song, Ms. Tracy C.M. Pang, Mr. Timothy C.K. Chu and Mr. W.L. Ma. My gratitude is also expressed to Mr. Tony M.K. Yung and Mr. C.K. Yu for their technical support. The Hong Kong Observatory is acknowledged for the supply of Hong Kong weather data.

The acknowledgement also includes my employer, the City University of Hong Kong, that had partially funded this research study, and those colleagues who had shared my administrative duties during my thesis-writing period.

Finally, I wish to dedicate this thesis to my parents who have given the greatest care to their children with love.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

List of Symbols

BB	building block
C_{ij}	amortised cost of (i, j) pipe segment
CDF	cumulative distribution function
CLI	cooling load intensity
$CPUL_{ij}$	amortised cost per unit length of (i, j) pipe segment
CV	Characteristic Vector
$c(e)$	capacity for each edge
$c(T_{best,f})$	best known solution
DCL	district cooling load
DCS	district cooling system
d	diameter of T
$d^g_{x_i,x_j}$	genotypic distance between the corresponding genotypes x^g_{xi} and x^g_{xj}
$d^g_{\min}$	minimum distance between two neighbouring genotypes
$d^p_{x_i,x_j}$	phenotypic distance between the phenotypes x^p_{xi} and x^p_{xj}
$d^p_{\min}$	minimum distance between two neighbouring phenotypes
$d\left(p^T_{i,j}\right)$	weight of the unique path from node i to node j in the spanning tree
E	a finite set of edges for connecting the nodes of a connected graph
FoLS	frequency of local search
FS	Finkelstein-Schafer statistic
f	friction factor

$f_k(x)$	objective function
G	connected graph
GA	Genetic Algorithm
GLS	genetic local search
h_f	head loss
k_B	Boltzmann's constant
L_{ij}	pipe length of (i, j) pipe segment
LB	Link Biased
LLS	Looped Local Search
LNB	Link and Node Biased
l	pipe length
MBE	mean bias error
m	population size
NB	Node Biased
NetDir	Direct Tree Representation
NetKeys	Network Random Keys
n	number of nodes
OCST	optimal communication spanning tree
OTTV	overall thermal transfer value
P_{succ}	success rate
Q	chilled water flow rate
RMSE	root-mean-square error
S	feasible region in the decision space
$S_n(x)$	value of the cumulative distribution function at x

Std Dev	standard deviation
Succ R	success rate
T	set of all vectors corresponding to the spanning trees in graph G
TMM	typical meteorological month
TMY	typical meteorological year
t_{conv}	generation number at which the best solution is obtained
V	a finite set of nodes
W	corresponding cost of edge E
w_k	weight representing as the relative importance of an objective compared to other objectives
WS	weighted statistics
x_{ij}	$\in$ {0, 1} is the decision variable x_{ij} = 1 if a pipe link (i, j) is connected x_{ij} = 0 if a pipe link (i, j) is not connected

Greek letters

δ_k	the absolute difference between the long-term CDF and the candidate month CDF
μ	mean value
σ^2	variance
α	weighting factor for annual pumping cost
ΔE	difference in energy level

CHAPTER 1

INTRODUCTION

The purpose of creating a thermally comfortable environment is to satisfy people's desire to achieve thermal neutrality in which neither warmer nor cooler surroundings would be preferred. Fanger (1972) has conducted outstanding research on the area of human thermal comfort in the past decades. Four major environmental factors, namely dry-bulb air temperature, relative humidity, mean radiant temperature and air velocity that can influence the condition of thermal comfort were identified; and their relationships with the heat exchange between human body and the surrounding were investigated. In the last century, scientists and engineers had spent great effort, by means of science and technology, to build controlled environment in which people can achieve a heat balance with the environment as well as thermal comfort. And this is the primary function of an air-conditioning system.

Who is the "Father of Air-Conditioning", Willis Haviland Carrier or John Gorrie? (Will, 1999). No one can give an authoritative answer to this question. However, there is a point without argument that the work from both of them opened the gate to modern air-conditioning engineering, from the early "comfort cooling using ice" to modern central air-conditioning system.

As the rapid development of densely populated urban areas, the highly packed building clusters with high thermal loads facilitate an environment under which a new form of "central" air-conditioning system has been developed and it is named as District

Cooling System that will be further described in the next section.

1.1 Description of District Cooling System

A district cooling system (DCS) is a scheme of cooling energy distribution through mass production. Cooling medium such as chilled water is generated at a remote and central refrigeration plant and then supplied to a district area comprising multiple buildings, through a closed-loop piping circuit.

A DCS scheme typically consists of groups of parallel chillers, heat exchangers, parallel or series pumps, piping network, in association with control valves, temperature/flow sensors and controllers, as shown in Figure 1.1. Each chiller is equipped with its associated constant-speed primary pump and the two are switched on and off together. For the distribution circuit, there are three major categories of chilled water pumping scheme options:

(i) Centralised pumping: the same water pump is used to distribute the chilled water through the whole system; a two-way control valve is installed at each distribution branch for regulating the chilled water supply to each building zone.

(ii) Distributed pumping: apart from the primary pump associated with each chiller, a distribution pump is provided at each distribution branch; the pressure control valve and the balancing valves are not required.

(iii) Primary-secondary pumping: a decoupler system comprises a production loop (DCS plant) and a distribution loop (district); the secondary variable-speed pumps are responsible for overcoming the pressure losses incurred by the chilled water

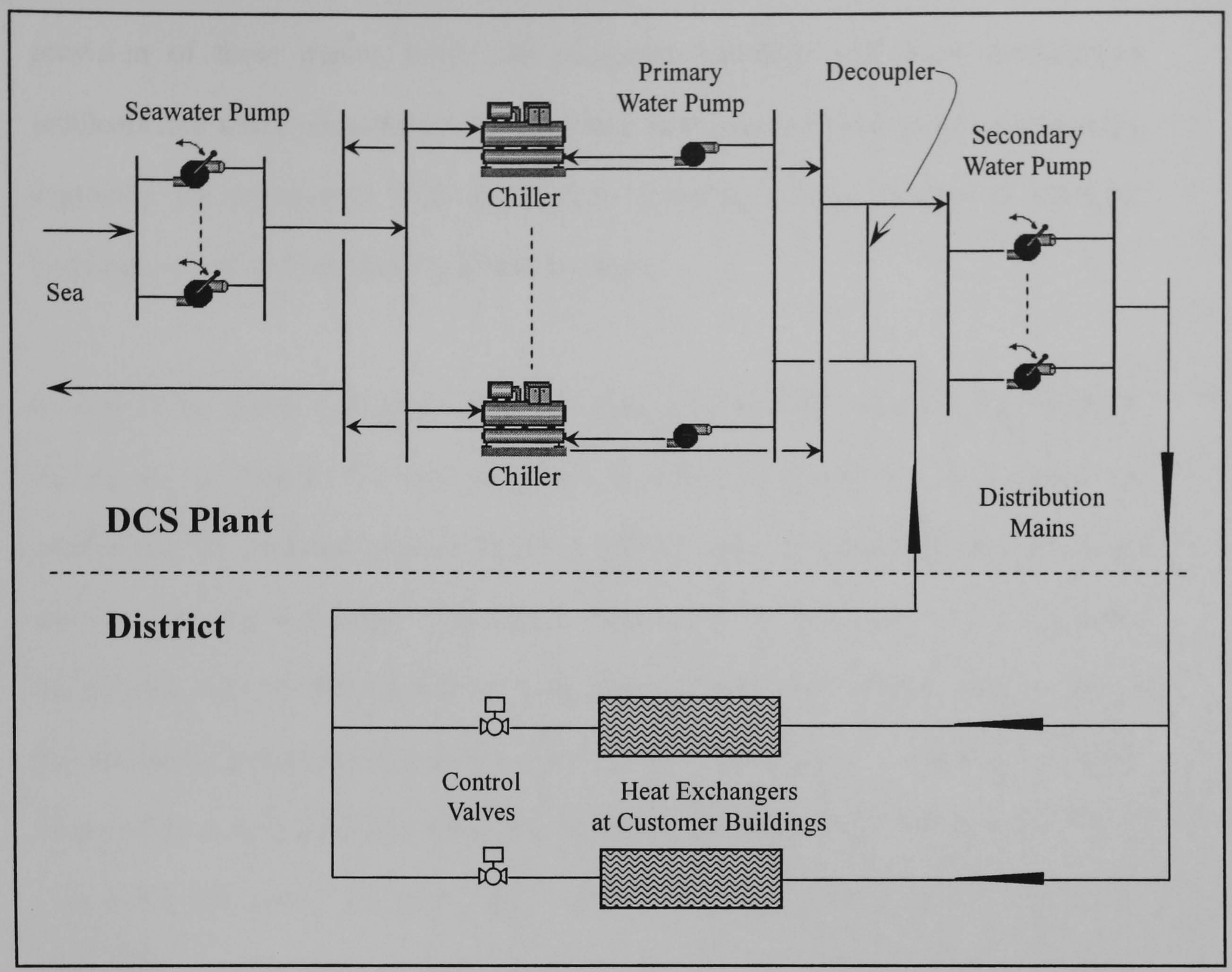


Figure 1.1 Schematic diagram of a district cooling system with direct seawater cooling

flow and maintain a critical pressure-differential in the distribution circuit.

The main advantages of the centralised pumping system are the low installation and maintenance costs and the simplicity of operation. However, this is not suitable for a DCS scheme of large scale since the pump head required for lengthy pipe work can be significant and the chillers need to withstand a high working pressure, which will result in a high energy cost. From the energy saving point-of-view, the distributed pumping system is the best option. However, this requires a number of secondary pumps to be scattered throughout the district and will increase the maintenance workload. Also the

provision of these pumps inside the consumer buildings will incur management problems like space allocation, right of access, additional investment on small pumps, especially for commercial DCS applications involving a large number of customer buildings owned and operated by different parties.

In view of the above, a primary-secondary pumping system is commonly used in DCS. As shown in Figure 1.1, the secondary distribution pumps are responsible for overcoming the pressure losses incurred by chilled water flow between the DCS plant and the consumer buildings. The distribution loop is hydraulically de-coupled from the primary loop by the presence of a decoupler bypass pipe between the two loops. The secondary pumps are all variable speed variable flow pumps. A control valve will be provided at each building branch for regulating the chilled water flow rate to be fed to each building zone. Cooling energy exchange between the distribution loop and the local chilled water loop of each building zone is via a heat exchanger. Indirect connection is commonly used as the interface between the DCS provider and the DCS customer buildings. Heat exchangers are installed to separate the chilled water circuits of the main and the users. This can avoid cross-contamination and over-pressurising the chilled water system for the buildings close to the central plant. The mass flow rate of the circulating chilled water is minimised by widening the difference in the supply and return chilled water temperatures. The pipeline is insulated to minimise the distribution energy losses.

The method of heat rejection in the DCS has a major implication on the efficiency of the chiller. Seawater cooling can result in the highest overall efficiency of the chiller plant. This requires the chiller plant to be located not too far away from the sea front and the

site for the construction of a seawater pump house with associated mechanical filtration and chemical treatment equipment. Another commonly selected option is the cooling tower for which measures should be taken to control legionnaires' disease, including correct biocide dosing, proper maintenance and cleaning schedule (Meesters, *et al.*, 2003).

1.2 Benefits of District Cooling System

The overall system efficiency of a DCS is higher than the individual chiller plants installed at single buildings. This is achieved, firstly, through the mass-scale production in which larger chillers with higher efficiency are in use, and secondly, through thermal load diversity in which the installed cooling capacity at the central chiller plant can be smaller than the total capacity to be installed at the customer buildings. Moreover, DCS customers can utilise their building space more effectively since the installation of their own cooling facilities is no longer required. From the environmental point of view, pollutant emissions and wastes (like CFC) from a remote site of district cooling plant are more easily taken care of than those released from small and scattered cooling plants all over the district. The above economical and environmental benefits are best experienced in a modern city where the cooling load density is high – typically in association with tall buildings.

There are many DCS installations found in different countries over the world, such as Europe, USA, Japan and some Asian countries. In Europe, the application of DCS is increasing steadily covering 14 countries (including France, Germany, UK, Sweden, Norway, Portugal, Italy, etc.) with about 70 cooling plants (Lam, 2004). Although there is a colder climate in those European countries, district cooling is well developed

and has been so for a long time. In the USA, the first commercial DCS has been in operation since 1962 at Hartford with a cooling capacity of 52 MW (Vadrot & Delbès, 1999). Since this project was established, district energy utilities have grown substantially. Now, there are over 6,000 district heating and cooling systems in the USA which provide 360,000 MWh of energy (in many applications, DCS is integrated with district heating plant to form a District Heating and Cooling (DHC) system) (Lam, 2004). Most of the output is used in institutional systems which serve groups of buildings owned by one entity, such as college, university, hospital or military base.

Japan is one of the pioneers in the field of DCS. The first application was constructed at the 1970 EXPO in Osaka of capacity 130 MW. The growth in number of DHC locations has increased significantly over the past 30 years. Currently, there are more than 220 DHC installations which mainly serve office buildings, where air-conditioning is needed, primarily for thermal comfort and the operation of computers. In other Asian countries such as Malaysia, Korea and China, the market for DCS continues to grow with climate change and increased standard of living. The major driving forces for developing DCS around the world are air pollution reduction and energy saving. It is recognised that DCS has created a win-win situation for building owners, users and society as a whole.

1.3 Current Technologies and Potential Area for Exploration

To establish a DHC scheme, a considerable amount of land space and intensive investment are required for setting up the central plant, constructing the piping network, and building auxiliary installations. In order to make the DHC scheme feasible and to maximise its benefits to the consumer buildings, a range of inter-related thermal

technologies, operation planning and piping arrangement for DHC have been investigated by various researchers.

(i) Inter-related Thermal Technologies

The implementation of integrated thermal technology associated with district energy system can increase the system energy efficiency and reduce the pollution-burden on the environment. Burer, *et al.* (2003) undertook multi-criteria optimisation of the design and operation of a district cogeneration plant integrating a superstructure which included a solid oxide fuel cell - gas turbine combined cycle, a compression heat pump, a compression chiller and/or an absorption chiller and an additional gas boiler. A Pareto-front was obtained as the global solution for the minimal cost associated with a given pollutant-emission abatement commitment.

The influence of a district heat accumulator on the economic performance of a cogeneration plant was assessed by Bogdan and Kopjar (2006). The plant supplies hot water for district heating, steam for industry and electric power. Economic benefits were achieved by charging the accumulator during the day time, when the electricity price was high; and by releasing district heat during cheap night hour, when other parts of equipment might be shut down. Both CO_2/SO_x and total annual electricity production were reduced.

The technical and economical feasibility of coupling thermal energy storage (TES) with cogeneration and DCS was evaluated by various researchers (Bahnfleth & Joyce, 1994; Hasnain, 1998; Khan, *et al.*, 2004). It was proved that cogeneration coupling with TES could maximise the utilisation of co-generated chilled water and offer a reduction

in both peak demand and energy consumption. Bernotat and Sandberg (2004) investigated the potential of applying biomass pellet fired small-scale cogeneration in combination with local heating networks for clustered dwellings. The outcomes showed promising results for clustered areas with good potential for small-scale district heating and combined heat and power worth further investigation. Henning, *et al*, (2006) studied the influence of policy and waste management on the operation of a district heating system. The results indicated that a wood fired cogeneration plant was more profitable than a natural-gas-fired combined cycle.

For a combined heating, cooling and power system of a DCS constructed in Beijing of China, Fu, *et al.* (2001) studied the significant effect of the supply and return water temperatures on the network where supply water of high temperature drives absorption chillers for air-conditioning in the summer, satisfies space heating demands in the winter and provides domestic hot water using heat exchangers throughout the year. The application of ice-slurry to a district cooling system was investigated by Kitanovski and Poredos (2002). The investigation showed that the slurry flow might present a Newtonian and non-Newtonian fluid, depending on the ice concentration and velocity. As the ice concentration increases and velocity decreases, the ice slurry behaves as a non-Newtonian fluid which can be used in district cooling system for safe operation.

(ii) Operation Planning

With good operation planning for a DHC system, optimal utilisation of the plant with minimum running cost can be achieved. Sakawa, *et al.* (2003) formulated an operation planning problem of a DHC plant as a nonlinear 0-1 programming problem. A solution was derived by an interactive fuzzy satisficing method through genetic

algorithms. Then an optimal operational plan was determined to minimise the cost of gas and electricity under the condition that the demand for cold water and steam could be met with minimum surplus. Benonysson, *et al.* (1995) applied a node-method to simulate the flow and temperature development of a given district heating system as a consequence of the consumers' heat loads and supply temperature from the plant. It was demonstrated that the optimal supply temperature could be determined with this approach to minimise the operational costs of the district heating system.

A mixed integer programming model was developed by Dotzauer (2003) to plan the production of heat and power for periods of up to one month for a district heating system. The study included the operation of fuel storage and the influence of the national tax system. The main output results were the power produced and consumed each day of the planning horizon which was important information for the hedging activities performed in the financial power market. A similar study was carried out by Söderman and Pettersson (2006). Mixed integer linear programming was used to formulate the structural and operational optimisation problem of a distributed energy system. In the model, production and consumption of electrical power and heat, power transmission, transport of fuels to the production plants, transport of water in the district heating piplines and storage of heat were taken into account. The solution gave a minimum total cost consisting of investment cost and running cost which was calculated with cumulative thermal or electrical energy for each period and for each unit part in the distributed energy system.

(iii) Piping Arrangement

Optimal design of piping arrangement can benefit a DCS with best thermal efficiency

and minimum cost. In order to achieve the minimum steady-state rates of heat loss from, or heat gain by the supply pipe for a district heating or district cooling system, experiments have been performed by Babus'Haq and Probert (1987) to determine the optimal configuration for horizontal double-pipe district heating/cooling distribution networks. In both district heating and cooling cases, having the warm pipe placed above the cold pipe in either a cold or hot rectangular trench led to lower steady-state rates of heat loss or heat gain than that with any side-by-side arrangement or with the cold-above-warm pipe arrangement. This finding was further confirmed by numerical analysis using finite-element and finite-difference techniques (Babus'Haq, *et al.*, 1990).

Emphasis on the optimal pipe size (diameter) of the pipelines was also addressed by various researchers. Phetteplace (1995) developed a rational design method, using branch-and-bound technique, that yields the optimal pipe sizes of the piping networks for district heating systems of moderate size with case-specific parameter values. The method allows for the inclusion of all major costs and constraints on the design of a realistic district heating network. Other approaches for the optimisation of pipe sizes were applied and investigated by different researchers such as linear programming (Altinbilek, 1981; Calhoun, 1971), nonlinear programming (Lansey & Mays, 1989; Ormsbee & Contractor, 1981), genetic algorithms (Dandy, *et al.*, 1996; Savic & Walters, 1997) and simulated annealing (Loganathan, *et al.*, 1995).

(iv) Potential Area for Exploration

As recognised, a major constraint in introducing a district cooling scheme is the high investment cost for the distribution network. In a district, how the piping network connects the consumer buildings would affect the lengths of the pipe segments with

different diameters, as well as the pressure loss along the pipe work. The piping connection can be radial, tree-like circuit or a mix of the two. Optimal configuration can result in both minimum piping initial cost and minimum pumping energy cost.

For optimal design of the piping network in a DCS, an exhaustive approach has been applied by some researchers (Bloomster & Fassbender, 1983; Chen, 2004) to preliminarily study and determine the optimal sizes of the water pipes; alternative layout designs; locations of various fittings; and operation of the system. Only limited numbers of alternative design schemes were simulated by the simulation model, especially for searching the optimal layout design option. Walters and Smith (1995) have applied a genetic algorithm to a tree network of water supply system for optimal layout geometry determination. The network is limited to nodes with equal spacing and fixed flow rates across the network. Ahn (1993) developed an outer flow search algorithm – inner optimisation procedure to modify an existing New York water distribution system for extended water services. The same New York water distribution system was used by Zhang (1999) with genetic algorithm coupled with a transient hydraulic simulation model to generate and evaluate trial pipe network modifications in search of an optimal solution. However, little work has been done on the optimisation of piping configuration in DCS. There is a potential in this area to conduct a detailed and systematic study on the optimisation of distribution piping configuration in DCS which can facilitate a successful implementation of the district cooling scheme.

For optimisation of the distribution piping network in DCS, the objective is to search an optimal piping configuration that minimises the infrastructure (piping) cost compatible

with the minimum pumping energy cost. An effective and efficient algorithm for the study of optimisation of the pipework layout design in DCS should be developed. Moreover, the major factors involved in the optimisation of pipework network in DCS are also investigated and identified.

In evaluating one of the objective functions, pumping energy cost, it is required to determine the hourly building thermal load as well as the chilled water demand of each building category through building thermal energy simulation. For precise building energy prediction, weather data is one of the key factors for successful building energy simulation. Currently, in Hong Kong, the weather year 1989 is accepted as the Test Reference Year (TRY) for use in building energy simulation. In TRY, 8,760 hours of climatic information for one year is selected by a simple procedure established by ASHRAE in 1970s (DoC, 1980). The principle of this selection procedure is to eliminate, in the order of importance, those years in the period of record containing months with extremely high or low air temperatures until only one year remains. For comparing the use of multiple-year measured weather data against several typical weather data sets, Crawley (1998) reported the simulation results of a prototype office building at eight different US locations, using the DOE-2E hourly energy simulation program. Due to the inherent weaknesses in the TRY selection method in which years in the period of record that have months with extremely high or low mean temperatures are progressively eliminated until only one year remains, this tends to result in a particularly mild year which excludes extreme conditions. A recommendation coming out of Crawley's study was to avoid the use of TRY, since no single-year data could fit well the typical long-term weather pattern.

Comparatively, the use of a synthetic year - Typical Meteorological Year (TMY) could serve better the prediction of building thermal load and energy consumption pattern. In 1997, ASHRAE undertook a research project 1015-RP for the development of International Weather Year for Energy Calculation (IWEC) weather files (ASHRAE, 2002). The data sets were completed in 2001 and contains TMY hourly weather files for 227 sites, all located outside USA and Canada (over 70 countries are represented). In total, there are nine cities of China included in the IWEC weather database. They are BEIJING, GUANGZHOU, HARBIN, KUNMING, LANZHOU, SHANGHAI, SHENYANG, URUMQI, and MACAU Special Administrative Region. Hong Kong, as one of the special administrative regions in China, was not involved in this project.

The mostly nearby city with TMY weather data is Macau. In deriving the TMY weather data for Macau, 18-year weather data (1982-1999) were used in the statistical analysis. Though measured data of solar radiation in Macau were available, the solar data in the project were derived using the Kasten (1981) model, for the reason of unifying the data processing procedures across all sites. The clear-sky irradiance outputs from the METSTAT (Meteorological/Statistical) clear-sky model (Maxwell, 1998) were fed into the Kasten cloudy-sky model. The cloudy-sky global irradiance outputs from the Kasten model were then fed to the Perez *et al.* (1990, 1992) models for the calculation of diffuse and direct radiation, and in turn, the diffuse and direct illuminance. There raised a concern of 'piling up' models in the generation of IWEC files (Thevenard & Brunger, 2001). The performance of the Kasten model often led to a very large dispersion in the estimated values of daily solar radiation, when compared to the measured data. Therefore, using the TMY weather file of Macau for research study in Hong Kong is not appropriate. It is necessary to develop a set of local TMY

data using measured weather data from Hong Kong.

1.4 Objectives of the Present Study

The objectives of this research work described in the coming chapters are:

(i) to develop a dataset of typical meteorological year for the study of optimisation of the pipework layout design in DCS;

(ii) to develop an effective and efficient optimisation algorithm to study the optimum pipework layout design in DCS;

(iii) to investigate and identify the major factors involved in the optimisation of pipework layout design in DCS.

1.5 Organisation of the Thesis

This thesis is organised into eight chapters, each dealing with a particular aspect of the research on the optimisation of the piping network for district cooling system.

Chapter 2 provides an overview of the research problems and explains the current development and understanding in this field. Various types of optimisation problem as well as different optimisation methods are reviewed. The current design approaches for the distribution piping configuration in DCS are presented. Then the problems surrounding the optimisation of piping configuration are discussed.

Chapter 3 presents a review of a number of different approaches for deriving weather data of a typical weather year and illustrates the details of the generation of a typical meteorological year (TMY) for Hong Kong using 25-year (1979-2003) weather data

measured by the Hong Kong Observatory. The typicality of the developed TMY weather file is evaluated by studying its performance through a series of building thermal energy simulations.

Chapter 4 focuses on the development of hourly building thermal loads for various building categories in a district using the dataset of TMY developed in Chapter 3. The optimisation problem in the present study is formulated and explained in detail.

Chapter 5 gives a detailed review on various types of encoding methods used in genetic algorithms. A detailed description of the research methodology with a selected encoding method for optimisation of the piping configuration in this study is presented. The advantages and approach of incorporating a local search technique into genetic algorithm is also discussed.

Chapter 6 studies the impact of various factors, through parametric experiments, on the GA performance for optimisation of the distribution piping network in DCS. The application of local search technique in GA for the optimisation is also investigated.

Chapter 7 describes the details of computational experiments using a genetic algorithm with the developed local search techniques on seven benchmark problems. Analysis of the experimental results in terms of solution quality and computational efficiency are presented.

Chapter 8 summaries the key research findings. The limitations of this study are explained and future research work is recommended.

CHAPTER 2

Background

The study of optimisation on various problems arises from human beings' strong will to pursue "as near perfect as possible". Different types of optimisation problems can be classified into two major groups, namely continuous and combinatorial optimisation problems. In this chapter, typical types of combinatorial problems such as the minimum spanning tree problem, travelling salesman problem and network flow problem will be outlined, then the features of the optimisation problem in distribution piping network of the present study as well as past related work will be presented. A review of various optimisation methods follows. The final section summarises the review and presents the planned work.

2.1 Classification of Optimisation Problems

Optimisation problems can be categorised by continuous or combinatorial variables (also known as discrete) (Haupt & Haupt, 2004). Continuous optimisation problems involve an infinite number of possible values such as the optimal set point of chilled water temperature in air-conditioning system; air-to-fuel ratio in internal combustion engine; portion of investment in various financial markets; etc. In combinatorial optimisation problems, there is a finite number of possible values. The optimal solution consists of a certain combination of variables from a finite pool of possible variables. In the following paragraphs, the definitions and characteristics of some common types of combinatorial problems are outlined.

Minimum Spanning Tree Problems

There are various types of combinatorial problems in engineering applications. One of the well-known types is the Minimum Spanning Tree (MST) problem. Consider a connected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots\ldots, v_n\}$ is a finite set of nodes and $E = \{e_1, e_2, e_3, \ldots\ldots, e_m\}$ represents a finite set of edges for connecting the nodes. Each edge has an associated positive real number denoted by $W = \{w_1, w_2, w_3, \ldots\ldots, w_m\}$ representing the corresponding cost. Let vector $\mathbf{x} = (x_1, x_2, x_3, \ldots\ldots, x_m)$ be defined as:

$$x_i = \begin{cases} 1, & \textit{if edge } e_i \textit{ is selected in a spanning tree;} \\ 0, & \textit{otherwise} \end{cases} \tag{2.1}$$

A MST of graph G can be expressed by the vector $\mathbf{x}$. Let T be the set of all such vectors corresponding to the spanning trees in graph G. Then the MST problem can be formulated as:

$$\text{Min } \{z(\mathbf{x}) = \sum_{i=1}^{m} w_i x_i \mid \mathrm{x} \in T\} \tag{2.2}$$

Under the umbrella of MST, there are new formulations which involve different types of MST problems such as capacitated MST problem (Garey & Johnson, 1979), degree-constrained MST problem (Zhou & Gen, 1997), diameter-constrained MST problem (Deo & Abdalla, 2000), stochastic MST problem (Ishii, *et al,*. 1981), probabilistic MST problem (Rohlf, 1978) and quadratic MST problem (Zhou & Gen, 1998).

(i) Capacitated Minimum Spanning Tree Problem

In the capacitated MST (cMST) problem, given a graph *G(V, E)*, a specified root node $v_o \in V$, a positive weight *w(e)* for each edge, a positive capacity *c(e)* for each edge, and a positive requirement *r(v)* for each node; the cMST problem requires to find a spanning tree *T* of *G* with the minimum weight among all spanning trees that satisfy the capacity constraint. The capacity constraint requires for every edge *e* in *T*, $c(e) \geq \sum_{v \in S(e)} r(v)$, where *S(e)* denotes the set of all nodes whose path to the root v_o in *T* contains edge *e*. One example of application is in centralised communication network design in which the root node is a central communication controller shared by many users. Each node in the graph sends a specified amount of flow towards the root node. The edge capacity is the flow allowed through an edge.

(ii) Degree-constrained Minimum Spanning Tree Problem

For the degree-constrained MST (dc-MST) problems, there are some degree constraints on each node such that, at each node v_j, the degree value d_j (the number of edges incident to each node) is at most a given value b_j. This type of MST problem can be formulated as:

$$\min\{z(\mathbf{x}) = \sum_{i=1}^{m} w_i x_i \mid d_j \leq b_j, \quad v_j \in V, \quad \mathbf{x} \in T\} \tag{2.3}$$

The dc-MST problem is a popular variation of MST which reflects the constraints in real-world design in terms of number of links connecting to a node. It arises in many practical situations including transportation networks connecting cities; back-plane wiring in circuit boards; and in telecommunication networks. In a road system, setting

the maximum number of roads to be allowed to meet at a crossing is an example of the degree-constrained MST problem.

(iii) Diameter-constrained Minimum Spanning Tree Problem

The diameter-constrained MST problem can be stated as: let $G = (V, E)$ be a finite undirected connected graph with a set of V of nodes and a set E of edges. Assume that a cost c_{ij} is associated with every edge $(i, j) \in E$, with $i < j$. A spanning tree of G is denoted by $T = (V, E')$, with $E' \subseteq E$. For every pair of distinct nodes $i, j \in E$, there exists a unique path P_{ij} in T linking i and j. The number of edges in P_{ij} is denoted by d_{ij} ; and $d = \max \{ d_{ij}: i, j \in V\}$ is the diameter of T. Given a positive integer $2 \leq D \leq |V| - 1$, the diameter-constrained MST problem is to find a minimum cost spanning tree T with $d \leq D$. The diameter-constrained MST problem arises in distributed mutual exclusion applications where message passing is used. For example, a logical spanning tree structure on a network of processors is imposed. Messages are passed among processors requesting entrance to a critical section and processors granting the privilege to enter. The maximum number of messages generated per critical-section is $2d$, where d is the diameter of the spanning tree. Therefore, a small diameter is essential for the efficiency of the algorithm. Minimising the edge weights reduces the cost of the network.

(iv) Stochastic Minimum Spanning Tree Problem

In the stochastic MST problem, the edge costs are not constant, but random variables. The objective is to find an optimal spanning tree satisfying a certain chance constraint. An example is the construction of a communication network connecting a number of cities. If the construction cost of line between one city and other city varies with time,

then they can be considered as random variables. The optimal connecting pairs of cities and budget are to be determined under a condition: the probability that the total cost exceeds budget is below a certain level. The problem can be formulated as:

Let $G = (V, E)$ denote undirected graph consisting of node set $V = \{v_1, v_2, \ldots\ldots, v_n\}$ and edge set $E = \{e_1, e_2, \ldots\ldots, e_m\} \subseteq N \times N$. Cost c_j is attached to each edge e_j.

Spanning tree $T = T(N, S)$ of G is a partial graph satisfying the following conditions:

(a) T has a same set of node as G.

(b) $S \subseteq E$

$|S| = n - 1$ where $|S|$ denotes the cardinality of set S.

(c) T is connected

T: $x_i = 1, \quad e_i \in S,$

$x_i = 0, \quad e_i \notin S.$

In the stochastic MST problem, the values of costs c_j's are not constant, but random variables. Therefore the stochastic MST problem is to minimise the cost function *(f)* subject to:

$$\text{Prob} \left\{ \sum c_j x_j \leq f \right\} \geq \phi \qquad (2.4)$$

$x_j = 0 \text{ or } 1, \quad X : \textit{spanning tree},$

where each c_j is assumed to be distributed according to the normal distribution

$N(\mu_j, \sigma_j^2)$ with mean μ_j and variance σ_j^2. They are mutually independent. The value of $\phi > 1/2$ is usually assumed.

(v) Probabilistic Minimum Spanning Tree Problem

For the probabilistic MST (p-MST) problem, it is the case that not all the nodes are deterministically present, but are present with certain probability. In a given weighted graph $G = (V, E)$, there is probability of presence $p(S)$ for each subset S of V. Given an a priori tree T, $L_T(S)$ is defined as the length of the tree which connects nodes from the set S of present nodes using only parts of T. With a cost function $c: E \rightarrow R$, and a probability function p: $2^V \rightarrow [0, 1]$, it is to find a tree T that minimises the expected length $E[L_T]$:

$$E[L_T] = \sum_{S \subseteq V} p(S) L_T(S) \tag{2.5}$$

where the summation is taken over all subsets of V.

In a communication network, nodes may represent communication centers. Edges represent the communication links and link costs are the communication costs among centers. The probability of failure p_i is the probability of blocked communication in center i. If the centers are blocked, they can only be used to establish communication between unblocked centers. Then the problem of finding a priori network structure of minimum expected cost is the p-MST problem.

(vi) Quadratic Minimum Spanning Tree Problem

The quadratic MST (q-MST) problem involves searching for the spanning tree of minimum cost under a quadratic cost structure, which means the interaction effects of the cost between pairs of edges while considering the spanning tree with minimum cost. There is presence of the interaction effects of the cost between pairs of edges. An example is transmitting oil from one pipe to another in which the cost may depend on the nature of the interface between two pipes. A similar pairwise interaction effect arises in the connection of overground and underground cables or in a transportation or road network with turn penalties. In all these cases, the presence of intercosts results in the minimum spanning tree problem with the quadratic, rather linear, objective, which is denoted as the q-MST problem. The problem can be formulated as: considering a connected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots\ldots, v_n\}$ is a finite set of nodes representing terminals or telecommunication stations and $E = \{e_1, e_2, \ldots\ldots, e_m\}$ is a finite set of edges representing connections between these terminals or stations. Each edge has an associated positive real number denoted by $W = \{w_1, w_2, \ldots\ldots, w_m\}$ representing the distance.

It is assumed that there is a cost c_{ij} associated with each pair of edges (e_i, e_j) where $i \neq j$ and also there is a direct cost w_i associated with selected edge e_i in the tree. The following MST problem with quadratic objective function is denoted as q-MST problem.

$$\min \{z(\mathbf{x}) = \sum_{i=1}^{m} \sum_{\substack{j=1 \\ j \neq i}}^{m} c_{ij} x_i x_j + \sum_{i=1}^{m} w_i x_i \mid \mathbf{x} \in T \} \qquad (2.6)$$

where c_{ij} denotes intercost and w_i represents length/cost.

$$x_i = \begin{cases} 1, & \textit{if edge } e_i \textit{ is selected}, \\ 0, & \textit{otherwise}. \end{cases}$$

Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is a classical combinatorial optimisation problem. It is relatively old and documented as early as 1759 by Euler (Michalewicz, 1996). Despite its history, TSP has become a target for the optimisation community. TSP is conceptually simple. The travelling salesman must visit every city in a certain territory exactly once and then return to the starting city. The objective of the TSP is to search a plan of itinerary for minimum total cost of the entire tour, with the given cost of travel between all cities. A TSP can be represented by a complete weighted and directed graph $G = (V, A, d)$ with n nodes. V is the set of nodes, A is the set of edges and d: $A \rightarrow$ IN a weight function associating a positive integer weight d_{ij} with every edge (i, j) that corresponds to the distance between cities i and j. For symmetric TSP, the distances between nodes are independent of the direction of traversing the cities, i.e. $d_{ij} = d_{ji}$ for every pair of nodes. However, in an asymmetric TSP, the distances are different, i.e. $d_{ij} \neq d_{ji}$. Typical applications include circuit board drilling, X-ray crystallography, instances arising in Very Large-Scale Integration (VLSI) electronic circuit fabrication, etc.

Network Flow Problem

Another class of combinatorial problem is network flow problem which lies at the cusp between several fields including applied mathematics, computer science, engineering and operations research (Ahuja, *et al.*, 1993). Typical applications involve electric power distribution network, highway and rail network, airline service network, computer network, etc.

(i) Minimum Cost Flow

The minimum cost flow problem is the most fundamental of all network flow problems. For each edge $(i, j) \in E$ having an associated cost, c_{ij} denotes the amount of cost per unit flow on that edge. The capacity u_{ij} is the maximum amount which can flow on the edge and a lower bound l_{ij} denotes the minimum amount of flow in that edge. The flow in an edge is represented by x_{ij}. The optimisation model can be formulated as:

$$\min \sum_{(i,j)\in A} c_{ij}\, x_{ij} \tag{2.7}$$

subject to the mass balance constraint where the total outflow of a node minus the total inflow of that node should be equal to the supply/demand of the node as shown below.

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \qquad \text{for all } i \in N \tag{2.8}$$

and flow bound constraint

$$l_{ij} \le x_{ij} \le u_{ij} \qquad \text{for all } (i,j) \in E \tag{2.9}$$

(ii) Shortest Path Problem

In the shortest path problem, the objective is to find a path of minimum cost/length from a specified source node *(s)* to another specified sink node *(t)* assuming that each edge *(i, j)* $\in$ A has an associated cost/length c_{ij}. Some of the applications in the shortest path problem are to find a path between two specified nodes of a network that has minimum length, or a path which takes least time to traverse, or a path which has the maximum reliability. The shortest path problem can also model a situation in which flow can be sent from a single-source node to a single-sink node in an uncapacitated network. That is, if *v* units of flow is sent from node *s* to node *t* and the capacity of each edge of the network is at least *v*, the flow would be sent along a shortest path from node *s* to node *t*.

(iii) Maximum Flow Problem

The maximum flow problem can be treated as a complementary model to the shortest path problem. In the shortest path problem, flow incurs a cost but is not restricted by any capacities. However, in the maximum flow problem, flow incurs no costs but is restricted by bounds. It is intended to seek a feasible solution which sends the maximum amount flow from a specified source node *s* to another specified sink node *t*. If u_{ij} is set as the maximum flow rate of edge *(i, j)*, the maximum flow problem is to identify the maximum steady-state flow which the network can send from node *s* to node *t* per unit time. Examples of the maximum flow problem are determining the maximum steady-state flow of vehicles in a road network or messages in a telecommunication network.

(iv) Transportation Problem

This is a special case of the minimum cost flow problem with the property that the node

set N is divided into two subsets N_1 (supply nodes) and N_2 (demand nodes). The edge *(i, j)* in E represents the distribution path from warehouse *i* to customer *j*, for each edge *(i, j)* in E, $i \in N_1$ and j $\in N_2$. A typical example of this problem is the distribution of goods from warehouses to customers in which the nodes in N_1 represent the warehouses and the nodes in N_2 represent the customers.

2.2 The Problem in the Present Study

In the present study, the objective is to search an optimal piping configuration of a district cooling system (DCS) with minimum sum of pumping and pipe work costs. A hypothetical site of reclaimed land with different building mix including offices, hotels, retails, schools, hospitals and mass transit railway station is proposed in the present study. All are considered as the potential DCS customers. The locations of the consumer buildings (nodes) are pre-fixed. The piping network is modeled by a graph with undirected links. Either a radial or tree-shaped network or mix of both are allowed. The objective is to find the optimal/near-optimal piping network configuration of a DCS that minimises the infrastructure (piping) cost compatible with the minimum pumping energy cost as listed in Equation (2.10) below.

$$\min \quad \left\{ Z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} C_{ij} x_{ij} + \alpha \sum_{k=1}^{8,760} Hourly\ Pumping\ Cost_k \right\} \tag{2.10}$$

subject to:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij} = n - 1 \tag{2.11}$$

$$\sum_{i \in U} \sum_{\substack{j \in U \\ j>1}} x_{ij} \leq |U| - 1, \qquad U \subset V, \qquad |U| \geq 2 \tag{2.12}$$

where

n = number of nodes in the piping network

C_{ij} = amortised cost of (i, j) pipe segment = $L_{ij} \times CPUL_{ij}$

L_{ij} = pipe length of (i, j) pipe segment

$CPUL_{ij}$ = amortised cost per unit length of (i, j) pipe segment which depends on pipe diameter

x_{ij} = $\in \{0, 1\}$ is the decision variable

$x_{ij} = 1$ if a pipe link (i, j) is connected

$x_{ij} = 0$ if a pipe link (i, j) is not connected

α = weight factor (value of 6.73 was used in the present study) (details are presented in Chapter 4)

Equation (2.11) is true for all feasible piping networks in which n nodes must have $n - 1$ links. Inequality (2.12) states that if there are more than $|U|$ - 1 links connecting the nodes of a subset U, this subset U will contain a cycle which is not allowed in this study.

There are two components in the objective function, namely the infrastructure (piping) cost and the pumping energy cost. The first one is closely related to a minimum spanning tree problem in which a finite set of edges for connecting the nodes can be defined. Each edge has an associated positive real number representing the corresponding independent cost. That is the piping cost of each pipe segment in the problem under the present study. The cost of each segment depends on the distance between the two connecting nodes; and the corresponding pipe diameter which is determined by the chilled water flow rate passing through the pipe segment.

The second part, pumping energy cost, is very much close to a minimum cost flow problem. In this network flow problem, each edge *(i, j)* has an associated cost which denotes the amount of cost per unit flow on that edge. The flow rate passing through each pipe segment cannot be determined until the whole piping configuration is formed. In the problem under the present study, the total cost of flow depends on both the total chilled water flow rate in the whole piping network and the pressure drop along the critical path. That means not the pressure drops in all the pipe segments will be taken into account. Only the sum of pressure drops along the critical path is used to calculate the pumping energy.

In the following paragraphs, current researches on similar optimisation problem are reviewed. For a study on the design and economics of district heating and cooling systems, Bloomster and Fassbender (1983) developed a computer simulation model, called GEOCITY. Within GEOCITY, several models are involved for specific purposes. A distribution system model allows user to input the layout design of a piping network for each district within a city and calculate fluid conditions in each segment of the piping network. Capital and operating cost models can be called to determine the cash flow for the construction and operation of the distribution system. The revenue to a reservoir model is the energy cost to the distribution system and is included in the cash flow for the distribution system. This simulation model can handle inflation and escalation in the economic calculations. From the distribution system cash flow, the required revenue and unit cost of the hot water/chilled water can also be determined.

GEOCITY has flexibility to simulate a wide range of system configurations with

various technical and economic parameters. A thermal storage system can be coupled to the district heating/cooling system. This simulation model can perform evaluation of the design and calculate the life-cycle cost. However, only limited numbers of alternative layout designs can simulated by this model.

A similar approach was used by Chen (2004) who constructed a simulation model for district chilled water and heating hot water distribution systems. During the process of building a model, information describing the network was collected and assembled. The author made use of a standard matrix solution technique to develop the simulation model. The method is H-Equation which solves, for each junction, by forcing continuity of flow through each connecting pipe. Simultaneously, the head loss across each pipe is updated based on the flow balance information. The flow rate and head are solved in an inner-outer loop algorithm where the flow is guessed. The head loss is calculated consistent with that guessed and the flow is updated according to the new pressure drop information. The Newton-Raphson method is employed to refine each successive solution, resulting in a sparse square matrix that is solved during each solution pass.

The simulation model was applied to two projects for district chilled water and heating hot water distribution systems. One was for the Texas A&M University Ross Street and the other was the campus of the University of Texas. For the first project, the objective was to determine the replacement of chilled water pipes using the original size (24-inch diameter) or a larger size (30-inch diameter). Simulations were conducted with focus on the pipe frictional loss as well as the building loop differential pressure. Moreover, the interruption on the continuous service in the campus of the university

under the replacement of chilled water pipes was also studied by simulation. The focus of the second project was to explore more cost-effective and robust design options. In the main campus of the University of Texas at San Antonio, there was a need to expand the central chilled water distribution system. Six different alternative layouts of the district heating/cooling system were designed and tested against each other by the simulation model. Through the simulation study, a design option with the lowest estimated construction and labour cost; and with the minimum interference to the university and utility plant operation was identified. Additional detailed simulation was conducted to investigate the optimal locations for placing the pipe fittings (tees and valves); and how to connect them to the existing central chilled water distribution system. The operation of a future central chilled water system by incorporating combined cycle cogeneration and district energy plant was also simulated with an aim to searching optimal operation in terms of low operational cost.

This was a component-based simulation approach. The author, based on a preliminary design, conducted detailed simulation to evaluate the optimal size of the pipes; alternative layout designs; locations of various fittings; and operation of the system. Again, only limited numbers of alternative designs were evaluated by the simulation model, especially for searching the optimal layout design option.

2.3 Review of Various Optimisation Methods

For the optimal design of a distribution piping system, using computer simulation for each case can only be applied to a limited number of design alternatives. The difficulty is due to the long computational time involved in the simulation for each individual case. An appropriate optimisation technique should be developed for

optimal design with acceptable solution quality and computational efficiency. In this section, a number of common optimisation methods will be reviewed.

(i) Exhaustive Search

Using the exhaustive search method, every solution in the search space has to be checked until the best global solution has been found (Michalewicz & Fogel, 2004). That means there is no way to be sure that the best solution has been found unless everything has been examined. This approach requires a long computational time to check every possible solution, especially for the size of the search space of real-world problems which is usually enormous. The favourable feature of the exhaustive search is that it is simple. There is no need for the researchers to design and construct any algorithm. The basic question to answer as to whether an exhaustive search is to be applied or not lies in the nature of the optimisation problem and the computational time involved and allowed. This method will be used when the simplicity of implementation is more important than speed. This is the case, for example, in critical applications where any errors in the algorithm would have very serious consequences; or when using a computer to prove a mathematical theorem. There are situations where it is sufficient to find a feasible solution that is close to optimal and it is not necessary to explore the entire search space. Different methods of exploring the search space might be preferable.

(ii) Greedy Algorithm

A greedy algorithm is an algorithm that follows the problem solving metaheuristic of making the locally optimum choice at each stage with the hope of finding the global optimum. It always takes the best immediate or local solution while finding an answer

but does not consider the details of possible alternatives. The general ideal behind greedy approach is that assigning the values for all of the decision variables one by one and at every step make the best available decision. Examples of greedy algorithm are Kruskal's algorithm, Prim's algorithm and Dijkstra's algorithm. The merit of greedy algorithm is its simplicity and often gives good approximations to the optimum. However, for most problems, greedy algorithm mostly fails to find the globally optimal solution because they make commitments to certain choices too early which prevent them from finding the best solution. One example is the nearest neighbour algorithm applied in travelling salesman problem in which the unvisited city nearest to the current city will be visited at each stage. The choice made by a greedy algorithm may depend on choices made so far but not on future choices or all the solutions to the subproblem. For each number of the cities, there is an assignment of distances between the cities for which the nearest neighbour heuristic may produce solution far from perfect. There is often a high price to pay for making greedy choices at the beginning of an optimisation process (Michalewicz & Fogel, 2004). Greedy methods are conceptually simple, but they normally pay for that simplicity by failing to provide good solutions to complex problems with interacting parameters.

(iii) Heuristic Algorithm

Heuristic algorithm is an algorithm for solving optimisation problems by the use of heuristics which is defined as serving to discover by trial and error. A heuristic is a method of performing a sequence of modification of a given solution in order to obtain a different solution. The actual modifications involve a neighbourhood search. The algorithm starts with an initial feasible solution which must be found by some specified methods. Main streams of the heuristic approaches including Local Search, Simulated

Annealing, Tabu Search and Genetic Algorithm are detailed below.

(iii)(a) Local Search

In optimisation process, instead of searching the entire space of possible solutions exhaustively, searching within some local neighbourhood of the current solution by local search is a useful compromise. It starts from a current solution and then iteratively moves to a neighbour solution. Typically, every candidate solution has more than one neighbour solution. The choice of which neighbour solution to move to is taken using the information about the solutions in the neighbourhood of the current one. If an improvement is found, the current solution is replaced by the improvement and the process is repeated. The algorithm stops when it reaches a locally optimal solution. The method of steepest descent searches the whole neighbourhood and selects the neighbour which results in the greatest improvement to the cost function of an optimisation problem. For random descent, it selects the neighbouring solutions randomly and accepts the first solution which improves the cost function. Local search is an attractive approach for many combinatorial optimisation problems as they have a natural neighbourhood structure.

(iii)(b) Simulated Annealing

The name, simulated annealing, comes from annealing in metallurgy. It is a popular method of escaping from locally optimal solution based on an algorithm in which a randomised neighbourhood search strategy is used. The technique involves heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat causes the atoms to become unstuck from their initial positions and wander randomly through states of higher energy. The slow cooling gives them more

chances of finding configurations with lower internal energy than the initial one. To simulate the evolution to thermal equilibrium of a solid for a fixed value of the temperature T, Metropolis *et al.* (1953) proposed a *Monte Carlo* method, which generates sequences of states of the solid. Given the current state of the solid, characterised by the positions of its particles, a small and randomly generated perturbation is applied by a small displacement of a randomly chosen particle. If the difference in energy, ΔE, between the slightly perturbed state and the current one is negative or equal to zero, i.e. $\Delta E \leqq 0$, then the process is continued with the new state. However, if $\Delta E > 0$, the probability of acceptance of the perturbed state is given by $\exp\left(-\frac{\Delta E}{k_B T}\right)$. k_B is Boltzmann's constant. This acceptance rule for new state is referred as the Metropolis criterion. Following this criterion, the system eventually evolves into thermal equilibrium. In statistical mechanics, this Monte Carlo method is known as the Metropolis algorithm.

By analogy with this physical process, simulated annealing algorithm was developed and applied in optimisation problems (Kirkpatrick *et al.*, 1983). Metropolis algorithm is used to generate sequences of configurations of a combinatorial optimisation problem. In that case, the configurations assume the role of the states of a solid while the cost function C and the control parameter c take the roles of energy and temperature, respectively. The simulated annealing algorithm can be viewed as a sequence of Metropolis algorithms evaluated at a sequence of decreasing values of the control parameter. Initially, the control parameter is given a high value. For a given a configuration i, another configuration j can be obtained by choosing at random an element from the neighbourhood of i. The latter corresponds to the small perturbation

in the Metropolis algorithm. Let $\Delta C_{ij} = C(j) - C(i)$, the probability for configuration j to be the next configuration in the sequence is given by 1, if $\Delta C_{ij} \leqq 0$. If $\Delta C_{ij} > 0$, the probability will be given by $\exp\left(\frac{\Delta C_{ij}}{c}\right)$. Thus, there is a non-zero probability of continuing with a configuration with higher cost than the current configuration. The control parameter is then lowered in steps. The process is continued until equilibrium is reached, i.e. until the probability distribution of the configurations approaches the Boltzmann distribution which is given by:

$$\Pr\{configuration = i\} = \frac{1}{Q(c)} \exp\left(-\frac{C(i)}{c}\right) \tag{2.13}$$

where $Q(c)$ is a normalisation constant depending on the control parameter c. The final frozen configuration is then taken as the solution of the problem at hand.

(iii) (c) Tabu Search

The basic idea of tabu search is that a memory forces the search to explore new areas of the search space. It is also designed for the purpose of escaping from local optima. Solutions that have been examined recently are memorised and they become tabu/forbidden points to be avoided in making decisions about selecting the next solution (Glover & Laguna, 1997).

The search procedures of a tabu search can be described as follows. It starts with an initial feasible solution S and selects a best solution among the neigbours of S obtained by non-tabu moves. Then the current solution is substituted by a solution selected as

above and the search in the neighbourhood is repeated. However, this procedure can cycle, i.e. visit some solutions more than once. In order to avoid this phenomenon, some identification criteria are introduced for moves which are expected to lead to cycles. Such moves are then added to a tabu list. On the other hand, forbidding certain moves could prohibit visiting interesting solutions. Therefore an aspiration criterion is introduced. It serves to distinguish the potentially interesting moves among the forbidden ones. The tabu list has a pre-specified length which may vary during the search. For maintaining the length of the tabu list within the pre-specified limits, some tabu moves have to be cancelled from the list. A first-in first-out rule is usually used to keep the length of the tabu list. The search procedures stop when a stop criterion, either a limited running time or a limited number of iterations, is fulfilled.

(iii) (d) Genetic Algorithm

Genetic Algorithm is another nature inspired approach to combinatorial optimisation problems. The underlying motivation of genetic algorithm is the attempt to borrow ideas from the selection process in nature which develops complex and well adapted species through relatively simple evolutionary mechanisms. Genetic algorithm is stochastic and population-based method (Bäck, 1996). The idea is that, given a population of individuals, the environmental pressure causes natural selection for survival. This causes a rise in the fitness of the population. Based on this fitness, some of the better candidates are chosen to seed the next generation by applying crossover and/or mutation to them. Crossover is an operator applied to two or more selected candidates (parents) and generates one or more new candidates (offspring). Mutation is applied to one parent and breeds one new offspring. A set of new candidates (offsprings) generated by crossover and mutation compete with the old ones

for a place in the next generation. In the evolution of each new population, probabilistic procedures are used so that the risk of the search collapsing onto a false optimum can be reduced. This process iterates until a termination criterion, e.g. solution of sufficient quality or computational limit, is met.

Local search as well as simulated annealing, tabu search are neighbourhood search based algorithms. Local search is similar to simple hill-climbing and the neighbourhood search is repeated until a locally optimum solution is found. However, there may be many locally optimal solutions that are not global optimal solution. The disadvantage of local search is that neighbourhood search based algorithms are improvement heuristics and thus highly depend on the starting solution. The search is only guided by local information. The choice of the starting solution in a neighbourhood search is extremely critical. An inappropriate choice may lead to a local optimum and thus high distance to the global optimum.

Simulated annealing differs in the acceptance criterion for neighbouring solutions. In local search, only better solutions are accepted in relation to the objective function value. With simulated annealing algorithm, it allows accepting solutions worse than the current solution. In order to successfully apply simulated annealing to combinatorial optimisation problems, a cooling schedule must be determined, which provides an initial value for the temperature as well as an updating rule for the temperature. However, there is no accepted methodology for choosing the schedule, since finding the optimal schedule is itself an optimisation problem. The main drawback is that there is no choice which will be good for all problems and there is no general way to find the best choice for a given problem (Groot, *et al.*, 1990; Kreher & Stinson, 1998).

An essential feature of tabu search is the use of memory. Forbidding moves in the tabu list can eliminate cycling. However, the maintenance of the tabu list and the searching within the list is often too time consuming to be practical. Moreover, if the tabu list is too long, there will not be very many allowable moves at any given point in the algorithm. This may make it difficult to find good solutions. The major drawback of this algorithm is the difficulty of tuning its control parameters including the length of the tabu list; and the number of the neighbours of the current solution which have to be scanned before updating the current solution. The dependence of suitable values of the control parameters on the problem instance is strong, but not clear. This may lead to a bad behaviour of the algorithm in terms of robustness (Cela, 1998; Michalewicz & Fogel, 2004).

Genetic algorithm is a population-based algorithm. The probability of choosing an unfavourable start configuration is minimised due to the selection of a large number of starting solutions distributed over the whole search space. The search is then focused on promising regions of the search space by successively narrowing the regions until the search converges. As stated by Eiben and Smith (2003), the evolutionary process is one of the two most powerful natural problem solvers (the other one being the human brain). In genetic algorithms, the genetic information about the best/good solutions of a population of solutions can give the search some directions (Wright, 1996). Genetic algorithms had been successfully applied by various researchers for solving combinatorial optimisation problems including water distribution systems (Walters, *et al.*, 1999; Smith & Walters, 2000; Babayan, *et al.*, 2006; Iancu, *et al.*, 2006; Lavic, *et al.*, 2007) and piping networks (Ito, 1999; Morley, *et al.*, 2001; Obara, 2006; Kessels, *et al.*, 2007).

For designing an optimal distribution piping configuration of a district cooling system, it is infeasible to apply an exhaustive search approach as there is an enormous number of different combinations of the piping arrangement. In this respect, the GA offers an advantage with its probabilistic search heuristics to perform a global sampling of the entire search domain for searching the optimal/near-optimal solution. The genetic information about the good solutions of a population of solutions can give good directions to the search. This type of heuristic algorithm for solving optimisation problem can find the near-optimal solutions within a reasonable amount of computational time. Therefore, in the present study, genetic algorithm was selected for the study of optimisation in the piping configuration of district cooling system.

2.4 Concluding Remarks

This chapter reviews the features of continuous and combinatorial optimisation problems. Examples of typical combinatorial optimisation problems have been outlined. A brief description of the optimisation problem in the distribution piping configuration under the present study is introduced. The outcomes of similar piping optimisation problems by other researchers have been reported. Due to the difficulty of the current design problem of distribution piping system in a district cooling installation, it is infeasible to apply exhaustive searching approach as there is an enormous number of different combinations of the piping arrangement. Therefore, the common types of optimisation methods have been reviewed. Because of the distinctive features offered by GA on its probabilistic search heuristics over the entire search domain, it was selected as the optimisation method in the present study.

Before applying a GA to the study of optimisation of piping network, a weather data file

of typical meteorological year was developed first. With this set of typical weather data, a full set of hourly building thermal load as well as hourly chilled water demand of each building sectors in a district was determined. Then an appropriate type of encoding method for representing piping configuration in GA was selected through detailed literature survey. With the hourly chilled water demand determined and suitable encoding method selected, a GA was applied for optimisation of the piping configuration in district cooling system. Through a series of computational experiments, the effects of various major factors and the application of local search in GA for the optimisation of distribution piping network was investigated. Moreover, in order to validate the effectiveness of the developed algorithm with the optimal settings, benchmark problems will be used for comparison. The details of the above-mentioned work are presented in the coming chapters.

CHAPTER 3

TYPICAL WEATHER DATA FOR BUILDING THERMAL LOAD SIMULATION

For the optimisation of distribution piping configuration, pumping energy cost is one of the major components in the objective function. In order to evaluate the pumping energy, it is required to determine the hourly chilled water demand of each building category through building thermal energy simulation. Weather data is one of the key factors for successful building energy simulation. Computer simulation software requires hourly weather data such as dry bulb temperature, dew point temperature, solar radiation, wind speed and direction, etc. Since weather conditions can vary significantly from year to year, there is a need to derive a set of typical weather year (TWY) data to represent the long-term typical weather condition over a year. In this chapter, a number of different approaches for deriving TWY have been reviewed and the generation of a typical meteorological year (TMY) for Hong Kong using 25-year (1979-2003) weather data measured by the Hong Kong Observatory will be presented. The typicality of the developed TMY weather file is then evaluated by studying its performance through a series of building thermal energy simulations.

3.1 Background

Weather data is one of the crucial inputs for precise building energy simulation. There are various types and formats of weather data available. The selection of an appropriate type depends on the objective of the study and the type of simulation software used. For instance, in the study of energy audit and survey, weather data of a

specific year will be used to evaluate the energy performance of a building against the actual energy consumption recorded. However, for comparative studies, a typical year representative of the long-term average of the climatic condition should be adopted. In the present study, a typical and representative weather year would be appropriate.

There are three major types of hourly weather data generally used in building energy simulation (Keeble, 1990).

(i) Multi-year Datasets: they include a representative range of weather conditions and are used for assessing the overall performance and sensitivities of a building design.

(ii) Reference Year: this is a single year of hourly data (8,760 hours) selected to represent the range of weather patterns that would typically be found in a multi-year dataset. It is intended to allow more economical simulation than multi-year datasets; and to form an equitable basis for comparing the predicted typical energy consumptions of different building designs.

(iii) Representative Days: they are hourly data for certain weather elements of some day types (e.g. clear/overcast sky or cold cloudy/cold sunny/hot sunny days) to represent the typical climatic conditions. Representative days are economical for simplified simulations.

For most of the building energy simulations, especially for a comparative study, weather data of a typical year is commonly adopted since it is computationally economic. Moreover, based on a consistent set of weather data, results from parametric simulations can be compared and analysed. Weather data from multi-year datasets cannot offer

these features while the data from representative days are too limited for precise simulations.

For deriving the weather data of a typical year, there are two major approaches, namely simulation-based and statistic-based. In the simulation-based approach, building energy simulations are carried out using weather datasets from various years. The year with weather data which results in energy performance closest to the long-term trend will then be identified and selected as the typical year. This approach has the limitation that it is too specific for the building type, building services systems and simulation software used in the selection process.

The alternative is a statistic-based approach which is based on statistical analysis of the weather data. This selection process is independent of the types of building, system or plant. After selecting a typical weather year, simulations will be carried out to evaluate the closeness of the selected weather data to the long-term climatic condition.

3.2 Test Reference Year and Typical Meteorological Year

There are two common types of typical year, namely Test Reference Year (TRY) and Typical Meteorological Year (TMY). In TRY, 8,760 hours of climatic information for one year is selected by a simple procedure established by ASHRAE in 1970s (DoC, 1980). The principle of this selection procedure is to eliminate, in the order of importance, those years in the period of record containing months with extremely high or low air temperatures until only one year remains. Extreme months are arranged in the order of importance for energy comparisons. Hot Julys and cold Januarys are assumed to be the most important. All months are ranked by alternating between the

warm half (May to October) and the cold half (November to April) of the year, with the months closest to late July or late January given priority. The first step in the selection process is to mark all the 24 extreme months. Then it is continuous to mark months, starting with the next-to-the-hottest July, then next-to-the-coldest January and so on, down the first column and then down the second column until only one year remains without any marked month. If two or more years remain without any marked month, the process is repeated with the third, fourth, etc, hottest or coldest extremes until only one year remains without any marked month. The remaining year is the TRY.

Hall *et al*. (1978) developed a method which is one of the most commonly accepted for generating typical weather years. The method used to select a TMY for a given location involves selecting, by statistical methods, one typical meteorological month (TMM) for each of the 12 calendar months from a period of years and concatenating the 12 months to form a TMY. This is done by comparing the cumulative distribution function (CDF) for each year with the CDF for the long-term composite of all the years in a period for four major weather indices including dry bulb temperature, dew point temperature, wind speed and solar radiation. The statistic used to measure the closeness of each year's CDF to the long-term composite for a given index is the Finkelstein-Schafer (*FS*) statistics (Finkelstein & Schafer, 1971). This statistic is the average absolute difference between the yearly CDF and the long-term composite CDF. The closer the two CDFs the smaller the value of *FS*. For each year, statistics computed for each index. A weighted sum, using various weighting factors, is calculated for the indices and this sum is used to select the potential candidates for the TMM. With this process, Hall *et al*. developed TMYs for 26 SOLMET Rehabilitation Stations in USA using hourly (or tri-hourly) weather data over a 23-year period

beginning in 1953 and extending through 1975.

TMY data sets for 239 meteorological stations, derived from weather data of years 1961-1990 were generated by National Solar Radiation Data Base (NSRDB) in 1994 (Marion & Urban, 1995). These data sets, based on more recent and accurate data, were recommended for use in place of the earlier SOLMET TMY data sets. In the development of these new weather data sets, the original procedures proposed by Hall were used. Modifications to Hall's method were made to better optimise the weighting factors of the weather indices as shown in Table 3.1. Moreover, an index for direct normal solar radiation was added. It was claimed that improvement in the comparison between annual direct normal radiation for the new TMYs and the 30-year annual average by about a factor of 2 was resulted (based on 20 geographically representative NSRDB stations). In order to distinguish between these two TMY data sets, the new TMY data sets were referred to as TMY2.

Table 3.1

Weighting factors given to weather indices in various TMY methods

Weather Index	**TMY (Hall, 1978)**	**TMY2 (Marion & Urban, 1995)**	**IWEC (ASHRAE, 2002)**
Max. dry bulb temp.	1/24	1/20	5/100
Min. dry bulb temp.	1/24	1/20	5/100
Mean dry bulb temp.	2/24	2/20	30/100
Max. dew point temp.	1/24	1/20	2.5/100
Min. dew point temp.	1/24	1/20	2.5/100
Mean dew point temp.	2/24	2/20	5/100
Max. wind speed	2/24	1/20	5/100
Mean wind speed	2/24	1/20	5/100
Total horizontal solar rad.	12/24	5/20	40/100
Direct normal solar rad.	-	5/20	-

In 1997, ASHRAE undertook a research project 1015-RP for the development of International Weather Year for Energy Calculation (IWEC) weather files (ASHRAE, 2002). The data sets were completed in 2001 and contains TMY hourly weather files for 227 sites, all located outside USA and Canada (over 70 countries are represented). Up to 18 years (1982-1999) of weather data were processed by using Hall's method. The composite indices used to select the most typical months applied their own sets of weighting factors, which are different to those used by Hall.

In total, there are nine cities of China included in the IWEC weather database. They are BEIJING, GUANGZHOU, HARBIN, KUNMING, LANZHOU, SHANGHAI, SHENYANG, URUMQI, and MACAU Special Administrative Region. Currently, the weather year 1989 is accepted as the TRY for use in building energy simulation for compliance with the Performance-based Building Energy Code in Hong Kong (HKSAR, 2003). However, there are inherent weaknesses in the TRY selection method. Firstly, it involves a process in which years in the period of record that have months with extremely high or low mean temperatures are progressively eliminated until only one year remains. This tends to result in a particularly mild year which excludes extreme conditions (Crawley, 1998). Moreover, the weather data of TRY do not necessarily represent the long-term mean. Building energy simulation runs using TRY weather data are noticeably less reliable in replicating average historical conditions (Huang, 1998). Compared to TRY, TMY can offer better performance in building thermal energy simulations. It was important that a TMY should be developed for the present study for precise evaluation of the cooling demand as well as the chilled water flow along the piping network. In the present work, Hall's procedures and IWEC weighting factors, which are in line with the international usage and giving equal importance to

dry bulb temperature and solar radiation, have been used to develop a TMY for Hong Kong.

3.3 Typical Meteorological Month Selection Procedures

The method used to select a TMY for a given location involves selecting, by statistical methods, one typical meteorological month (TMM) for each of the 12 calendar months from a period of years and concatenating the 12 months to form a TMY. For each of the 12 calendar months, the procedure involves selecting five candidate years which are closest to the composite of all the years under study. The statistical analysis begins with the generation of nine daily weather indices: daily maximum, minimum & mean dry bulb temperature and dew point temperature; daily maximum & mean wind speed; and daily total global horizontal solar radiation for each day of record. Each of the nine sets of the daily indices are sorted into bins by month and then are used to establish 12 long-term cumulative distribution functions (CDFs), i.e. one for each month, covering the whole period of record, as well as short-term CDFs for individual months in each individual years. The CDF for each weather index x is determined by a value of the CDF at x, $S_n(x)$, as follows:

$$S_n(x) = \begin{cases} 0 & for\ x < x_{(1)} \\ (k-0.5)/n & for\ x_{(k)} \le x \le x_{(k+1)} \\ 1 & for\ x \ge x_{(n)} \end{cases} \qquad (3.1)$$

where

n = total number of elements

k = ranked order number (k = 1, 2, 3, ..., n-1)

The Finkelstein-Schafer statistic *FS* is then used to measure the closeness between the corresponding short- and long-term CDFs as shown below.

$$FS = \frac{1}{m}\sum_{k=1}^{m}\delta_k \tag{3.2}$$

where

m = the number of non-repeating reading over the long-term period for the candidate month

δ_k = the absolute difference between the long-term CDF and the candidate month CDF at x_i

For each of the candidate months, the nine different statistics *FS* calculated for the nine indices are grouped into a composite weighted index using Equation (3.3) and the weighting factors as listed in Table 3.1. Each *FS* value is multiplied by a corresponding weighting factor *W* which emphasises the perceived importance of the corresponding index for building energy simulations. The weighted statistics *WS* is then defined as:

$$WS = \sum_{j=1}^{9} W_j (FS)_j \tag{3.3}$$

All the individual months are ranked in ascending order of the *WS* values. A typical month is then selected by choosing from the five months with the lowest *WS* values, i.e. the one with the smallest deviation from the long-term CDF. In Hall's original method, persistence structures characterised by frequency and run length of days were included.

The persistence of mean dry bulb temperature and daily global horizontal radiation are evaluated by determining the frequency and run length above and below fixed long-term percentiles. For mean daily dry bulb temperature, the frequency and run length above the 67th percentile and below the 33rd percentile are determined. For global horizontal radiation, the frequency and run length below the 33rd percentile are also determined. The persistence data are used to select, from the five candidate months, the month to be used in the TMY. The highest ranked candidate month in ascending order of the *WS* values that meets the persistence criterion is used in the TMY. This criterion is not applied in IWEC's approach since it has been shown that it can lead to the rejection of all the five candidate typical months, particularly when the data are modelled and only a small number of years of data are available.

3.4 Selecting TMMs and Generating TMY for Hong Kong

There are 12 TMMs coming from different years to form a TMY. In the present study, hourly measured weather data of years 1979-2003 were acquired from the Hong Kong Observatory. The weather data include dry bulb temperature, dew point temperature, wind speed, global horizontal solar radiation, relative humidity, wind direction, sky cover, atmospheric pressure and rainfall. The first four sets of weather data are used for selecting TMMs and, together with the remaining sets of data, to construct a TMY for Hong Kong in the IWEC format.

Daily maximum, minimum and mean values of dry bulb and dew point temperatures were derived from the hourly measured data. Similarly, daily maximum and mean values of wind speed, and daily total solar radiation were determined. The values of *WS* for each month of the 25-year period were then calculated and the results are shown

in Table 3.2. The five candidate years for each month with the lowest values of *WS* are shown in italic, whereas the lowest one is shown in bold. In our case, the IWEC's approach was followed. Hence the selection criterion of evaluating the persistence structures proposed by Hall's original method was not applied. From Table 3.2, it can be seen that the 12 TMMs are evenly spread in the 25-year period. Five TMMs are found from 80s, another four from 90s and the remaining three are found from 2000, 2002 and 2003, respectively.

The 12 selected TMMs were then used to form a TMY in accordance with the IWEC format (Chan *et al.*, 2006a). The major data elements include dry bulb temperature, dew point temperature, relative humidity, wind speed, wind direction, global horizontal radiation, direct normal radiation, diffuse horizontal radiation, atmospheric pressure and precipitable water. These data are available except for the direct normal and diffuse solar radiation, because only hourly global horizontal radiation has been measured by the Hong Kong Observatory since 1979. In the present study, the correlation model proposed by Li and Lam (2001) was used to estimate the hourly direct normal and diffuse components from the measured global horizontal solar radiation.

Table 3.2

Weighted sums of the *FS* statistics for each month of the 25-year period (1979-2003)

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
79	0.11178	0.13158	0.10355	*0.05569*	0.11603	0.10853	0.14844	0.10413	0.08616	0.25590	0.11677	0.09428
80	*0.05533*	0.12511	0.06561	*0.04828*	0.10667	0.09978	*0.05678*	0.07804	0.07768	*0.05077*	0.11092	0.07556
81	0.09671	0.09099	0.09181	0.11483	0.07788	0.10464	0.09316	0.08589	0.08976	*0.05834*	0.11294	0.13631
82	0.08533	0.08800	*0.05876*	0.10606	0.06669	0.09890	0.06697	0.05430	***0.03723***	0.07394	0.11061	0.08000
83	0.15086	0.21034	0.19288	0.08602	0.07753	0.08865	0.09646	0.06869	0.09972	0.14505	0.13020	0.07497
84	0.14171	0.13553	0.08745	0.13746	0.09874	*0.07072*	0.13599	0.05876	0.06643	***0.04005***	*0.05769*	0.09302
85	0.08427	0.13280	0.12502	0.10896	0.13141	0.11754	0.09058	0.06753	0.09019	0.06148	0.06962	*0.06869*
86	0.14279	0.10628	0.09240	***0.04732***	*0.05093*	*0.04860*	*0.05307*	0.05777	0.09230	0.06307	0.08386	0.08072
87	0.10165	0.13641	0.12964	0.06479	0.09265	0.07196	0.06316	0.11243	0.04993	*0.05754*	0.11585	0.09863
88	0.12340	***0.05989***	0.15209	0.10719	0.08676	0.09960	0.07307	0.10449	0.08927	0.08383	0.13441	0.09935
89	0.09163	0.09759	0.10843	0.08578	0.07245	*0.06984*	0.06270	*0.05028*	0.06052	*0.05311*	***0.04612***	*0.06777*
90	0.09447	*0.06824*	0.08560	0.09908	0.10255	***0.04078***	*0.05001*	0.11212	*0.04689*	0.08696	0.06918	0.09782
91	0.07341	0.08920	0.08950	0.07044	0.09943	*0.05221*	0.07089	*0.04954*	0.07556	0.06944	0.07750	0.11079
92	0.12087	0.10939	0.09649	0.07559	0.11100	0.07246	0.07003	0.09111	0.13106	0.13723	0.13308	0.10226
93	0.10983	0.13516	*0.04668*	*0.06082*	*0.04913*	0.07866	0.10491	*0.04764*	0.05452	0.08363	0.08715	***0.06130***
94	*0.06118*	0.10378	0.09013	0.16359	0.13927	0.08583	0.15151	0.11862	0.11446	0.11064	0.14007	0.18024
95	***0.04980***	0.09691	0.07798	0.07461	*0.05623*	0.08245	0.08923	0.15233	*0.04942*	0.09789	*0.06326*	0.10044
96	0.08666	0.08457	*0.04697*	0.12262	0.07745	0.10192	*0.05904*	*0.04648*	*0.04016*	0.06310	0.08089	0.07059
97	*0.05623*	*0.06065*	0.09459	0.08021	***0.04561***	0.09679	0.13350	0.07865	0.11320	0.11069	0.09700	0.12032
98	0.10869	*0.08012*	*0.04211*	0.14624	0.07921	0.11148	0.07936	0.12102	0.05878	0.06558	0.10882	0.07502
99	*0.05894*	0.17585	0.07563	0.12467	0.06979	0.10966	0.09359	0.08214	0.05393	0.07729	0.06497	0.07892
2000	0.07012	*0.06754*	0.10152	0.06105	*0.05777*	0.10231	***0.04266***	0.06475	0.12517	0.08403	0.08926	*0.06722*
2001	0.07833	0.10248	0.13473	*0.05440*	0.08643	0.09180	0.12720	0.07425	0.07398	0.09023	0.08017	*0.06666*
2002	0.06983	0.14664	0.14420	0.16261	0.11344	0.09197	0.09826	***0.04479***	0.07309	0.09816	*0.06076*	0.10138
2003	0.09180	0.13604	***0.04008***	0.08902	0.08446	0.08737	0.14170	0.05734	*0.04328*	0.06950	*0.06278*	0.11260

Note: (i) The weighted sums in bold and shaded cells show the TMM.

(ii) The weighted sums in italic show the lowest five values in the month.

TMM	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Year	95	88	2003	86	97	90	2000	2002	82	84	89	93

In order to remove the discontinuities created by concatenating TMMs from different years to form the TMY, the cubic spline curve-fitting technique (Ayyub & McCuen, 1996) was used. This technique was applied to the six hours each side of the month interfaces for smoothing the weather data against discontinuities.

As an initial checking of the weather dataset obtained from the TMMs, the monthly average values of the four major weather indices, i.e. dry bulb temperature, solar radiation, dew point temperature and wind speed were plotted in Figures 3.1 to 3.4. The three curves in each figure are respectively the TMY, the 25-year (1979-2003) average and the TRY (HK-1989) for ready comparison. In Figure 3.1, it can be seen that the variations of monthly average dry bulb temperature of both TMY and TRY are close to the 25-year average since both methods apply dry bulb temperature as the major index in selecting the typical months. However, for monthly average total solar radiation, as shown in Figure 3.2, the results of TMY are closer to the long-term average while those from TRY have relatively larger deviation from the 25-year long term distribution, especially in February, March and April. It is mainly caused by the use of one single determining weather index (dry bulb temperature) in the selection process of TRY so that the perceived importance of solar radiation cannot be reflected in the selected TRY. The variations of monthly average hourly dew point temperature are shown in Figure 3.3. It can be seen that the three curves are close, particularly for the warm and hot seasons from April to November. For wind speed, the weighting factor used in the IWEC approach is 10% while it is zero in TRY selection. In Figure 3.4, none of the two curves of TMY and TRY can be concluded as close to the long-term average, yet the TMY curve performs better than the TRY curve.

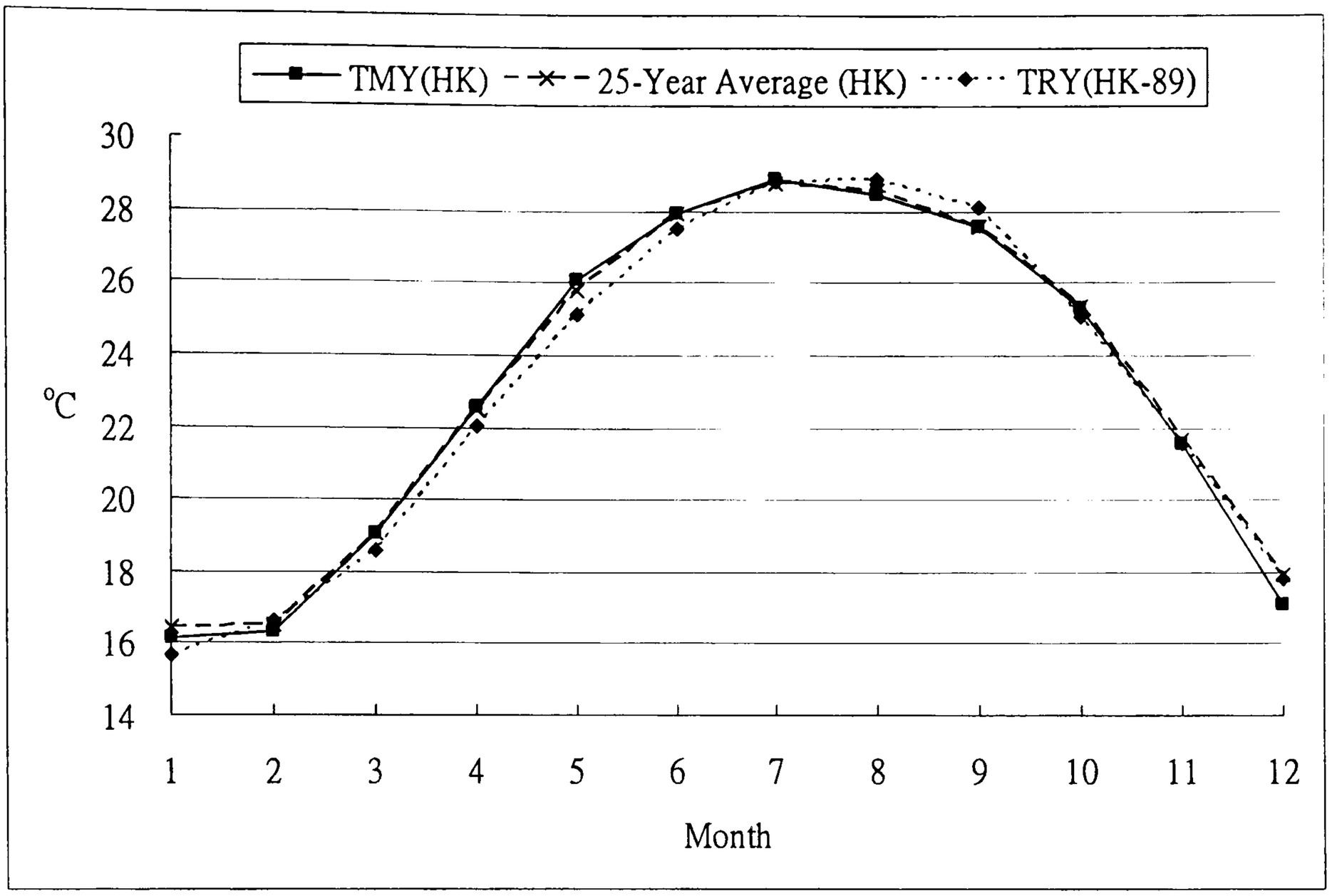


Figure 3.1 Monthly average hourly dry bulb temperature

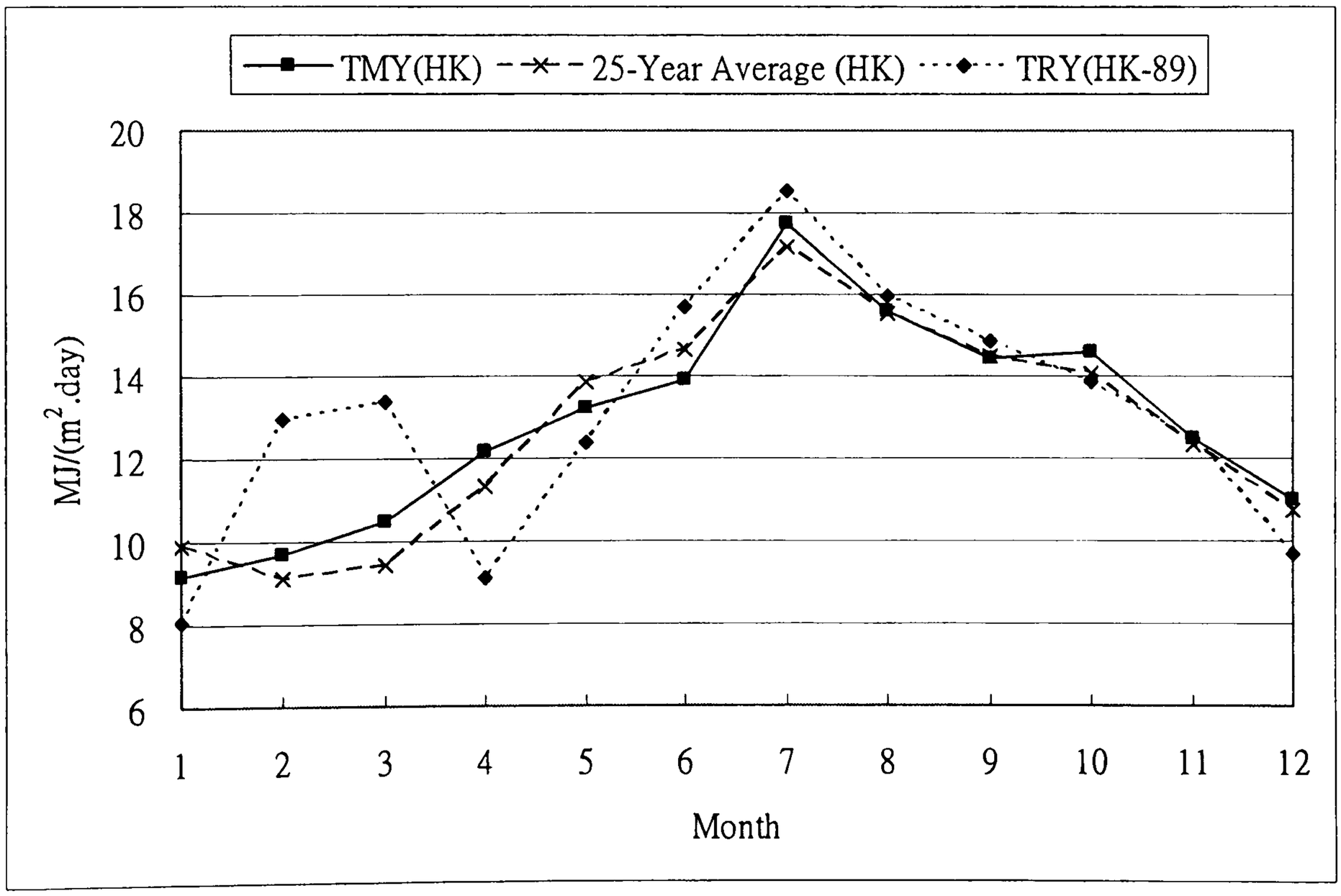


Figure 3.2 Monthly average daily total solar radiation

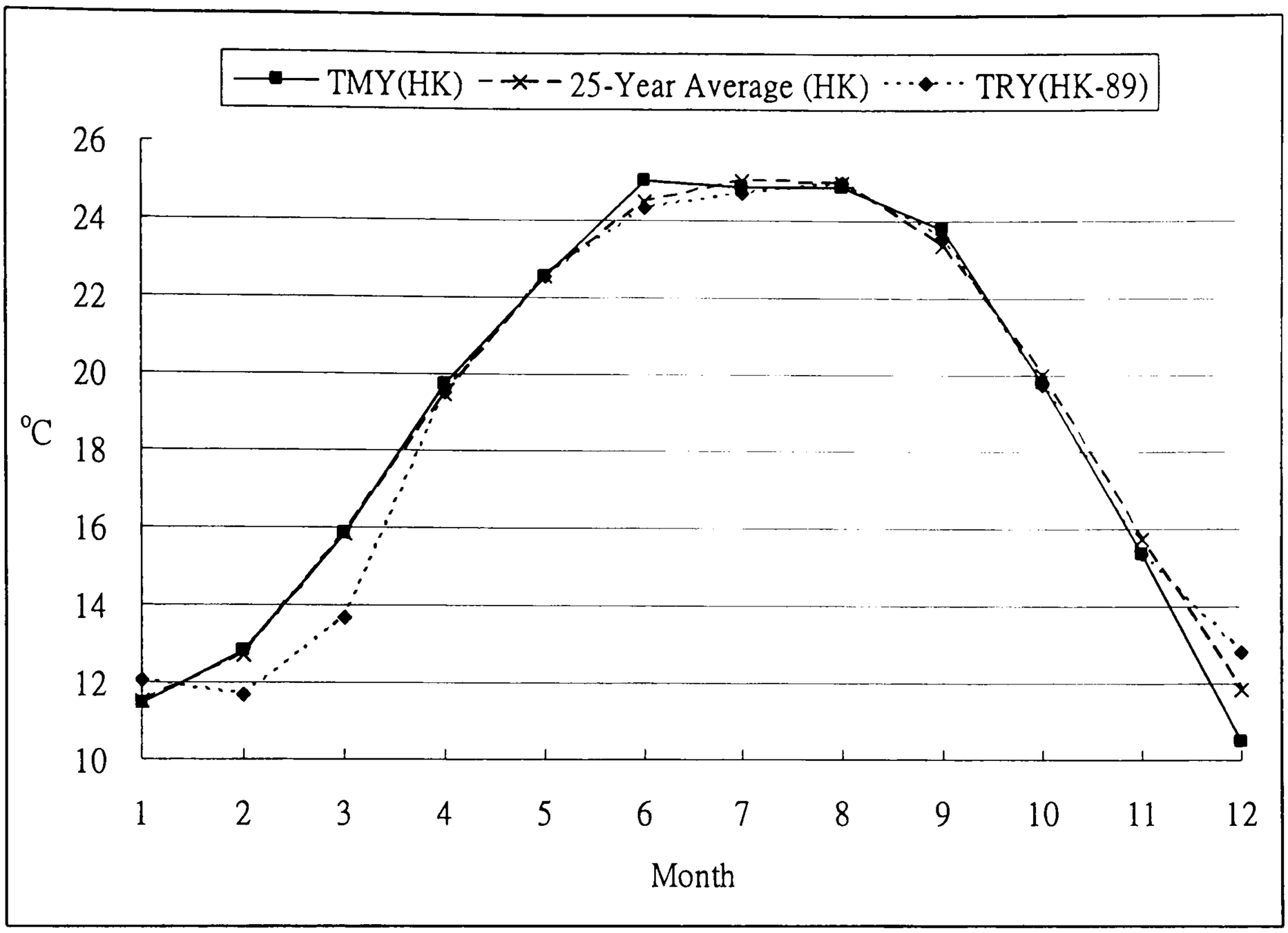


Figure 3.3 Monthly average hourly dew point temperature

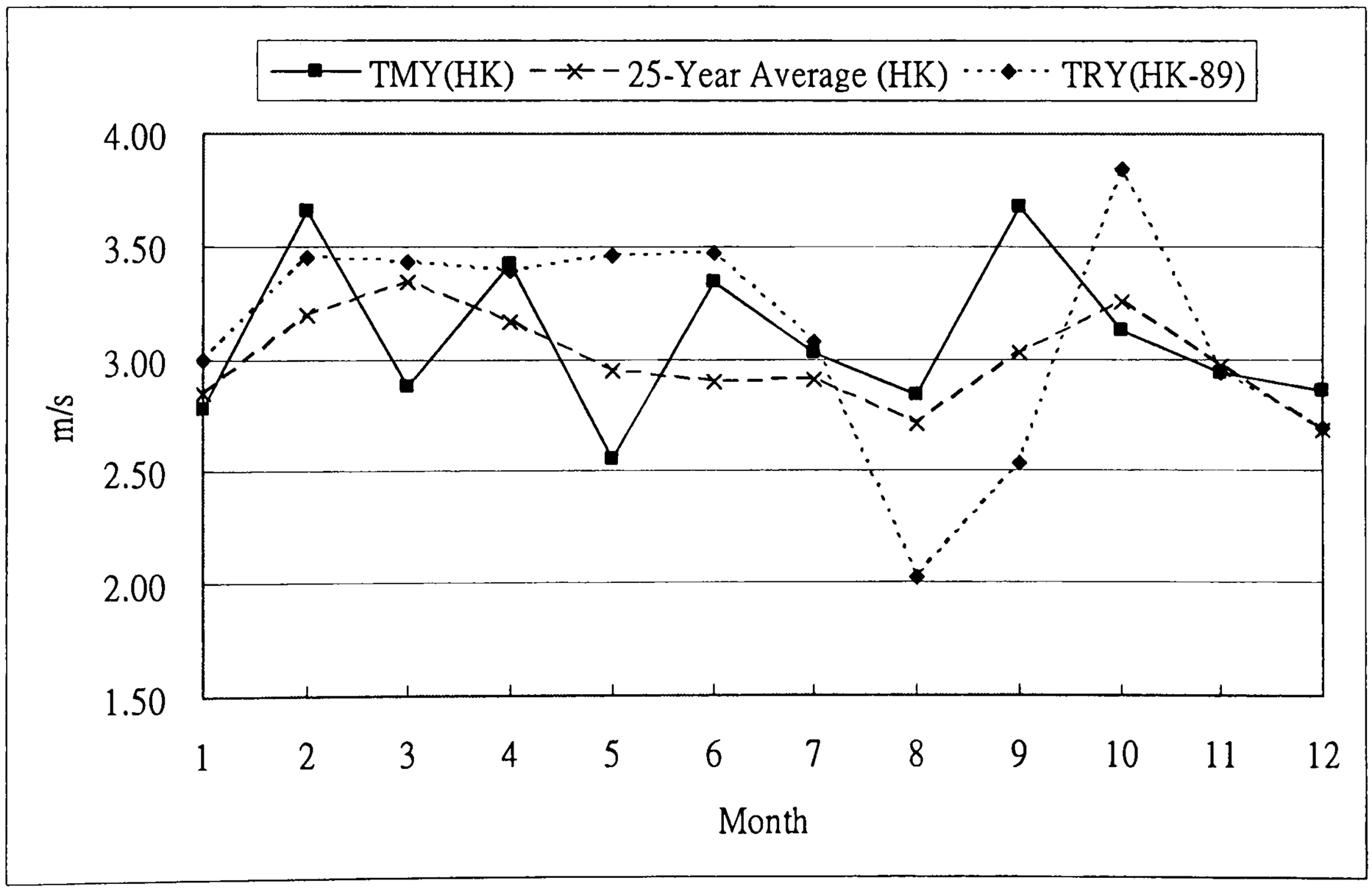


Figure 3.4 Monthly average hourly wind speed

3.5 Assessing the Typicality of the TMY Generated

For assessing the typicality of the TMY generated, computer simulations using weather data from the generated TMY and the individual years from 1979 to 2003 were performed as comparative study on the thermal performance and energy consumption profiles of a building.

The simulation tool used was EnergyPlus (DoE, 2005) building energy simulation computer program. For this comparative study, a generic office building was developed making reference to the local building energy code published by the Hong Kong Special Administrative Region Government (HKSAR, 2003). The generic building developed is a 30-storey office building (40m by 40m) with a centralised heating, ventilating and air-conditioning (HVAC) system. The floor-to-floor height is 3.6m and the window-to-wall ratio (WWR) is 60%. The building and the HVAC system operate on a 13-hour day (07:00 to 20:00) and a 5½-day week basis.

Figures 3.5 (a) – (e) show the twenty-five monthly electricity consumption profiles, produced by computer simulations, for the individual years from 1979 to 2003. It can be seen that all the years show similar seasonal patterns in the electricity use. Electricity consumption peaks during the six hot and humid summer months from May to October. Predictions from the TMY were also analysed and compared with the those from the twenty-five years. Figure 3.6 shows that the predictions from the TMY lie within the maximum and minimum range of predictions from the twenty-five individual years, and follow quite closely the 25-year long-term mean.

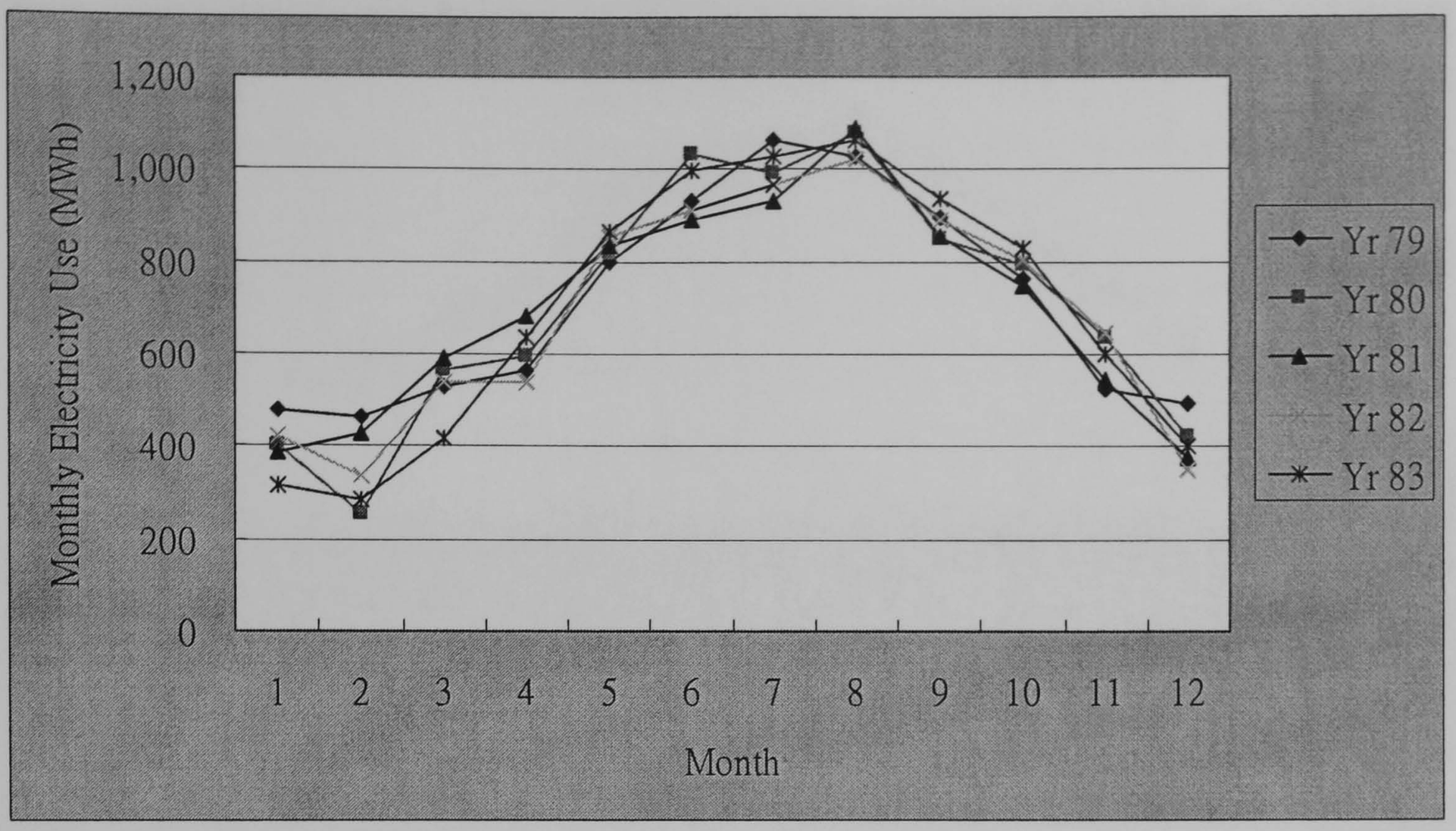


Figure 3.5a Predicted monthly electricity use from five individual years (1979-1983)

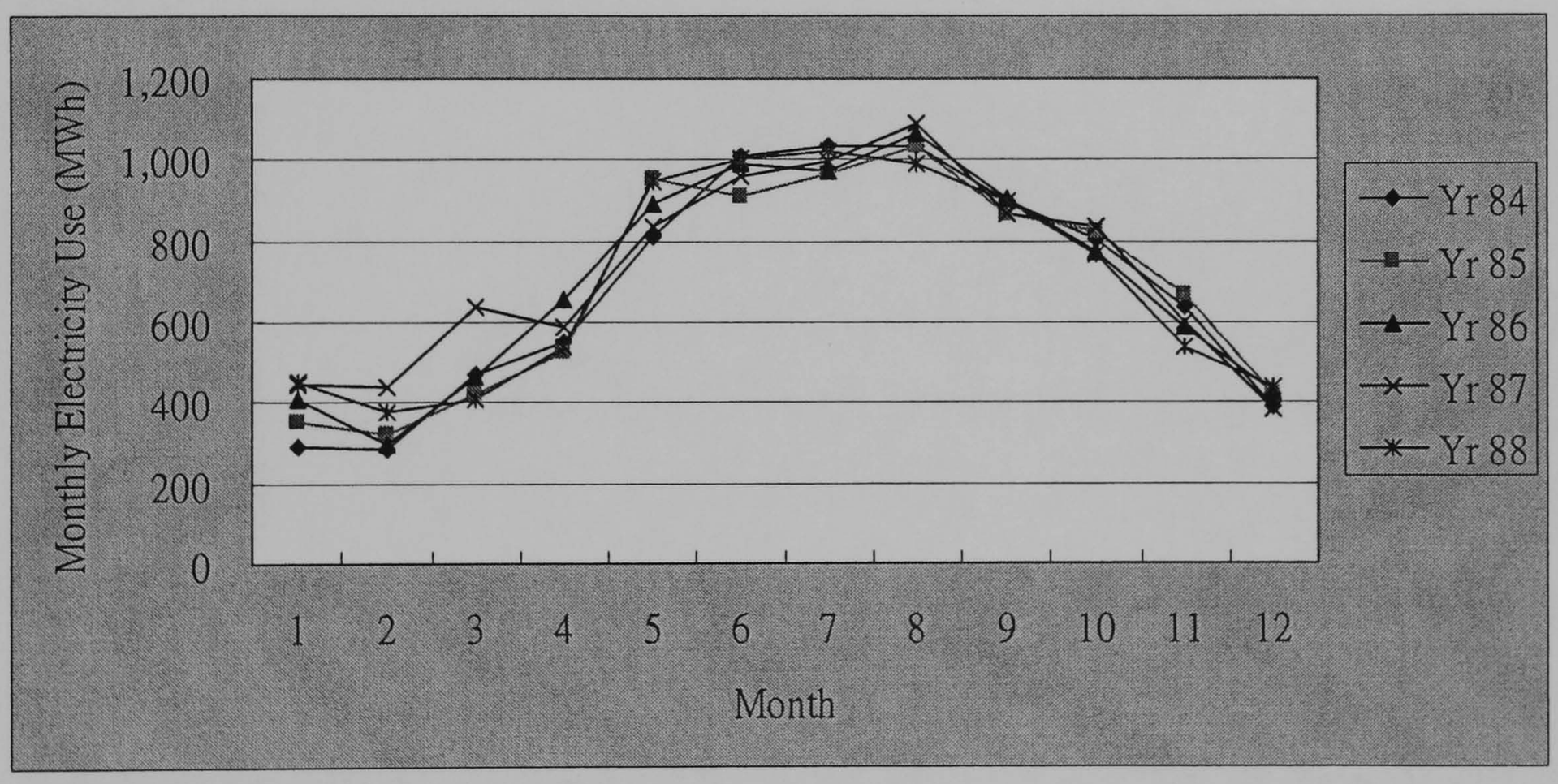


Figure 3.5b Predicted monthly electricity use from five individual years (1984-1988)

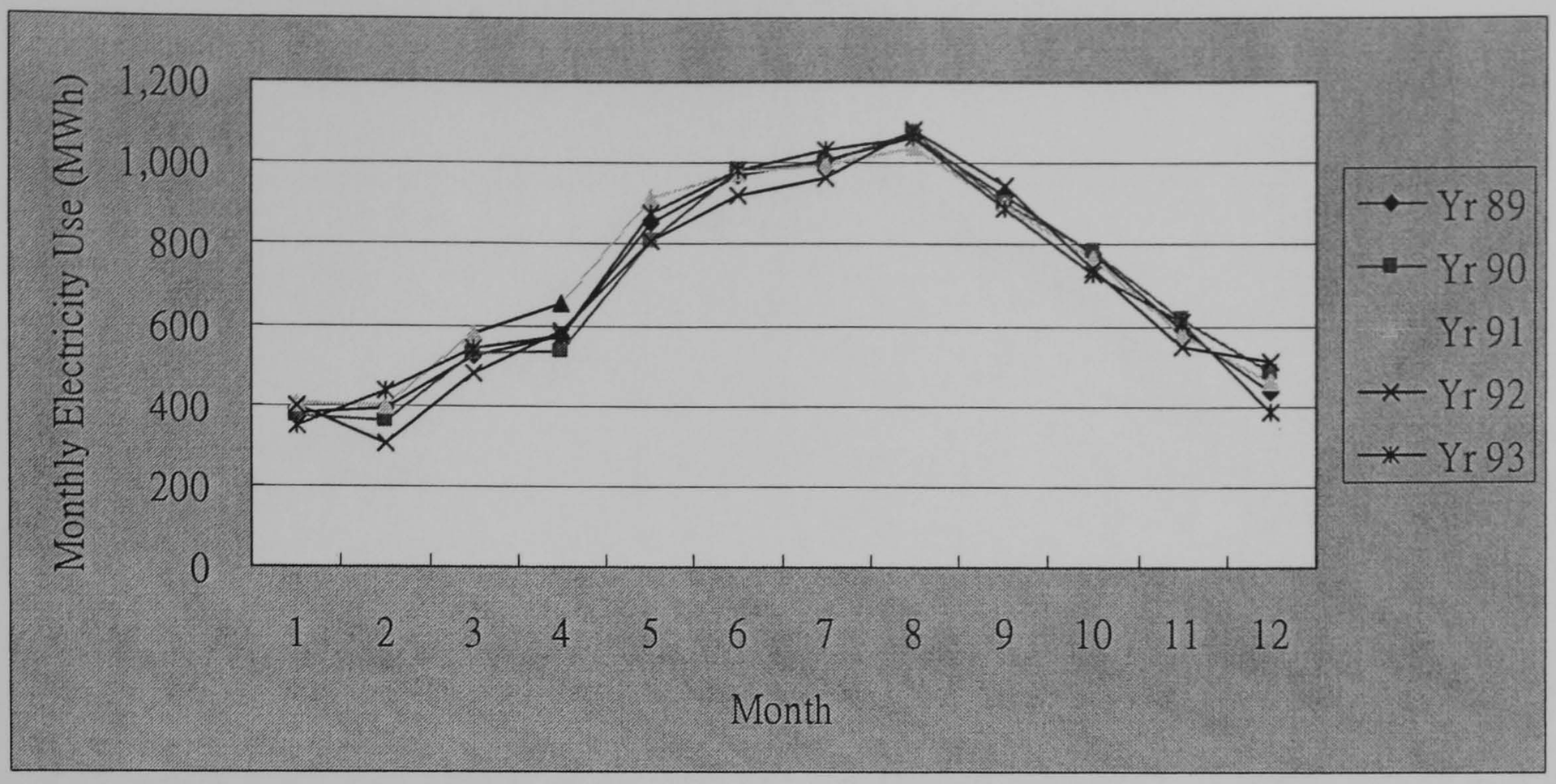


Figure 3.5c Predicted monthly electricity use from five individual years (1989-1993)

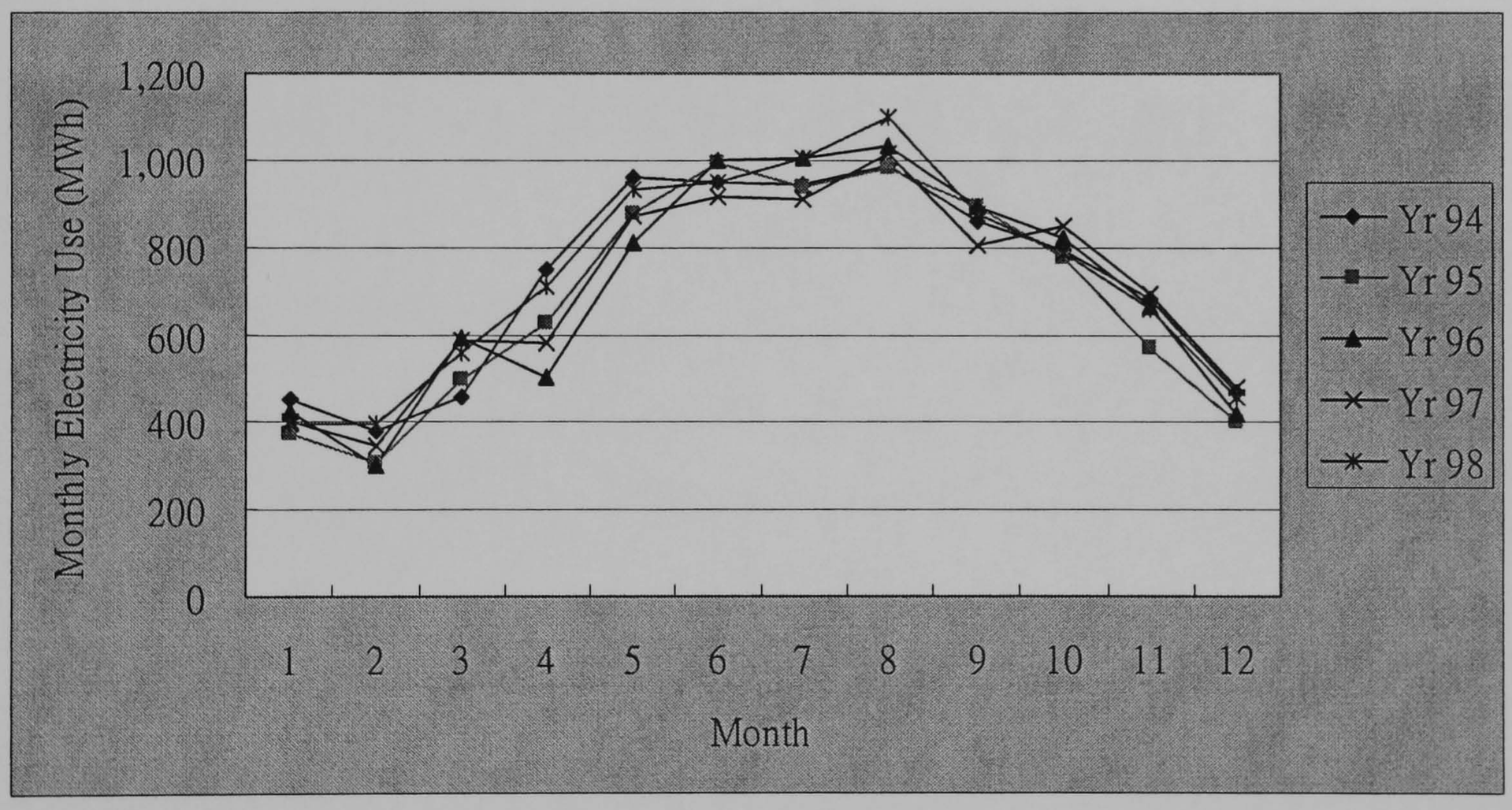


Figure 3.5d Predicted monthly electricity use from five individual years (1994-1998)

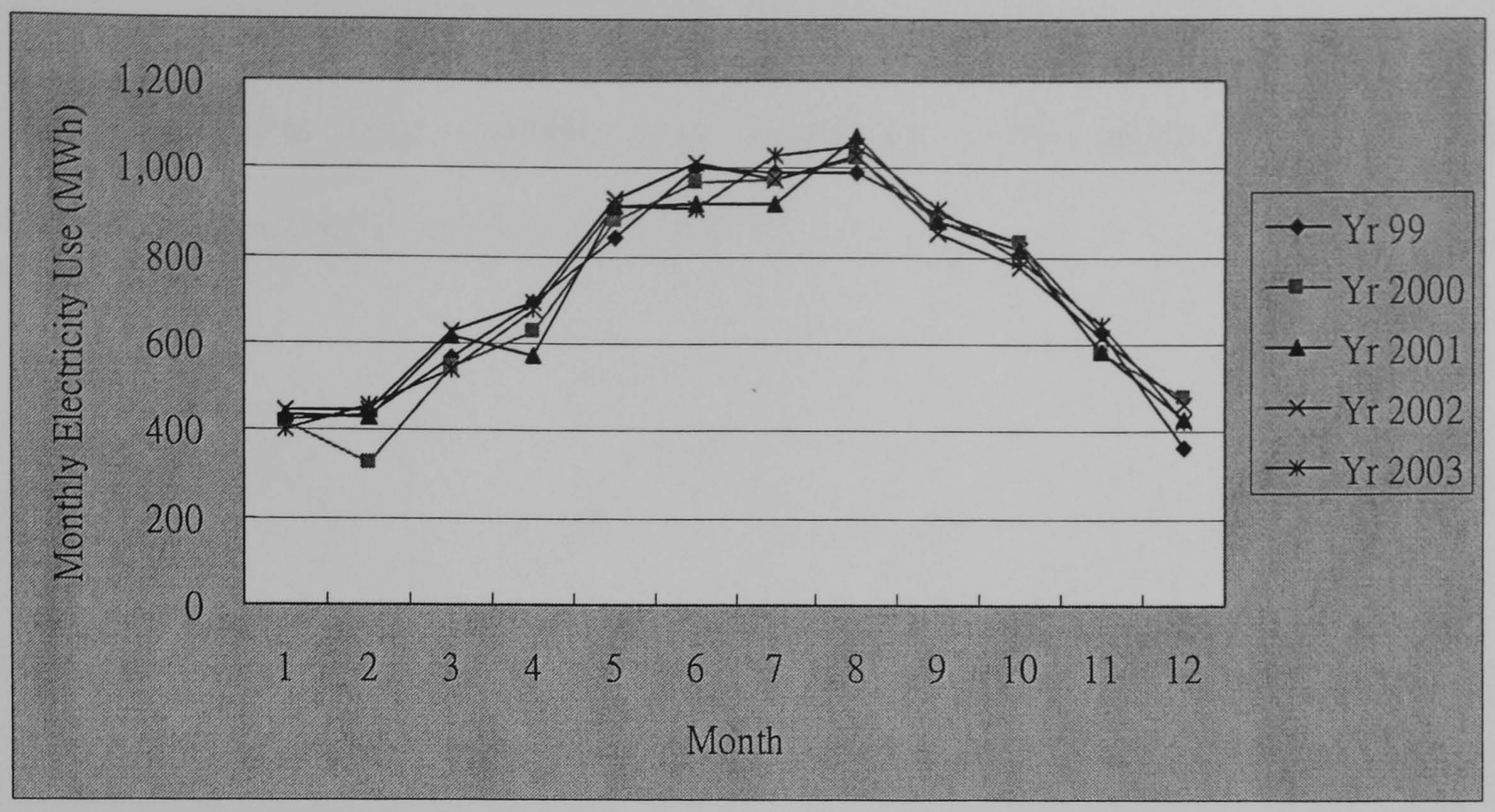


Figure 3.5e Predicted monthly electricity use from five individual years (1999-2003)

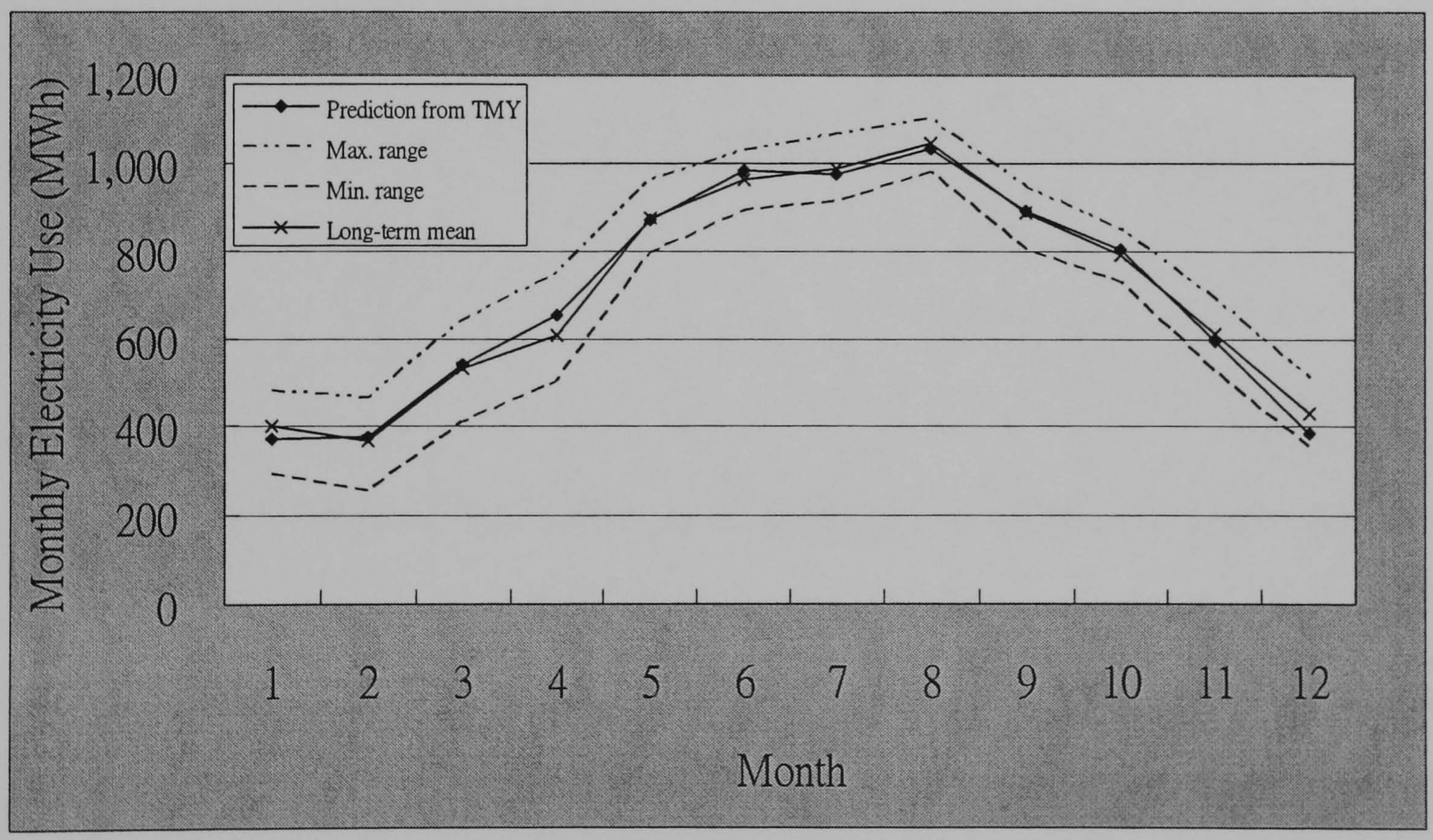


Figure 3.6 Predicted monthly electricity use from TMY

To compare the typicality of the different weather years, mean bias error (MBE) and root-mean-square error (RMSE) were calculated. The mean bias error and root-mean-square error are defined as follows:

$$MBE_j = \frac{\sum_{i=1}^{12}(x_{ij} - y_i)}{12} \tag{3.4}$$

$$RMSE_j = \sqrt{\frac{\sum_{i=1}^{12}(x_{ij} - y_j)^2}{12}} \tag{3.5}$$

where

x_{ij} = monthly electricity consumption for individual year from 1979 to 2003 (j = 1 to 25)

y_i = monthly electricity consumption for the 25-year long-term mean

$$= \frac{\sum_{j=1}^{25} x_{ij}}{25}$$

A positive MBE indicates that the annual electricity consumption is higher than the long-term annual prediction and vice versa, while the RMSE is a measure of how close the monthly profile is to the long-term. Figure 3.7 shows the MBE and RMSE for the twenty-five individual years and the TMY. The MBE ranges from -23.4MWh for 1984 to 31.9MWh for 2002. In terms of annual electricity use, there are 280.8MWh less than the long-term value of 8,490MWh and the annual consumption from 2002 is 382.8MWh more.

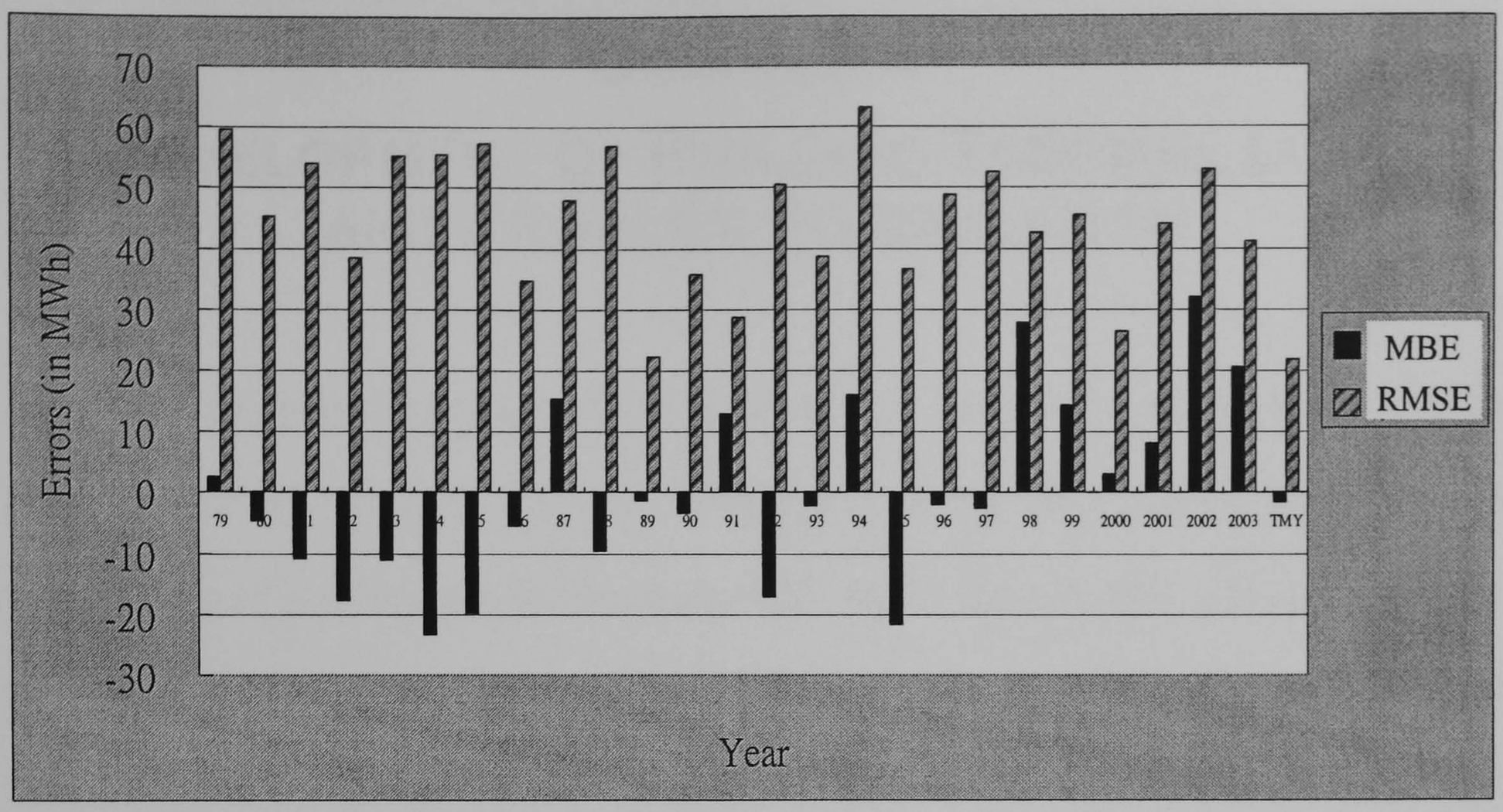


Figure 3.7 Mean bias error and root-mean-square error for the different years

The smallest value of MBE, -1.3MWh, was found in year 1989 and its annual electricity use is only 0.18% less than that of the long-term. The value of MBE for the TMY is -1.6MWh. The annual electricity use from the TMY is 8,470MWh which is 0.23% less than that of the long-term. Although the year 1989 has the smallest MBE, it is only the result of a fortuitous cancellation between over- and under-estimation. The closest monthly profile is the TMY which has the smallest RMSE of 21.8MWh, representing 3.08% of the long-term monthly mean consumption.

From the analysis of the predicted electricity consumption for different individual years, it is envisaged that the TMY developed in the present study can give a good indication of the long-term energy performance in building energy simulation and it will be used for developing building thermal loads for various types of building in a district; the details will be presented in the next chapter.

CHAPTER 4
DEVELOPMENT OF BUILDING THERMAL LOAD AND PROBLEM FORMULATION

To predict the periodic energy consumption of the distribution pumps in the piping network of a district cooling system, a data base of the space cooling loads of each specific type of building was firstly developed. In the present study, a hypothetical site of reclaimed land with different building mix was proposed. All were considered as the potential customers of a district cooling system. The hourly cooling load profile as well as the chilled water demand along each piping segment were determined using the weather dataset of the typical meteorological year developed in Chapter 3. In the following sections, the development of building thermal load and the formulation of the problem to be optimised are detailed.

4.1 Hypothetical Site and Building Grouping

A hypothetical site of reclaimed land which provides 461 hectares of land as a major waterfront development was proposed for the present study. A building mix including offices, hotels, retail, schools, hospitals, indoor game hall, museum and mass transit railway station were developed into this strategic area. All were considered as potential DCS customers. Town planning data including site area, population, types and distribution of the buildings was sought through surveys and consultations (TDD, 2000).

In 1998, the Territory Development Department (TDD) of the Hong Kong Special Administrative Region (HKSAR) Government commissioned a feasibility study for a

district area (South East Kowloon) to review and carry out a preliminary design for the early development works. The study included the preparation of layout plans and preliminary design drawings for infrastructure and civil engineering work and town planning data. Extracted from the inventory of planned premises and potential DCS users in this district development, the type, number, and distribution of the end users were identified. These end-user buildings were then grouped into a limited number of application categories based on their intrinsic characteristics and functions such as the similarity in site usage, occupancy periods, indoor controlled environment, fresh air rates and load diversities.

In this study, a total of eight categories of non-domestic building types were established and the building information is shown in Table 4.1. Category (1) includes general and hi-tech offices. Category (2) comprises the indoor-game hall, public swimming pools and related indoor sports facilities. Category (3) refers to primary and secondary schools. Hospitals in category (4) operate for 24 hours each day. Category (5) includes the cruise terminal and the aviation museum. Category (6) is solely for hotel guest rooms. The retail area in category (7) includes shopping malls allocated in both the residential and commercial developments. Category (8) refers to the passenger terminal of the mass transit system.

4.2 Developing Building Thermal Load

For each building category, one model building was developed to represent the building typically constructed in the district. The generic model building of each building category, with regard to the construction materials, physical size and facilities, was then developed. The internal load densities and the daily operating schedules for the three

Table 4.1

Building information of individual building in various categories

	Building Category	Gross Floor Area (GFA) (m^2)	No. of Storey -	Occ. Density (m^2/person)	Lighting Density (W/m^2)	Equipment Density (W/m^2)	Window to Wall Ratio (%)	Fresh Air Requirement (liters/second/person)	Indoor Design Criterion Dry Bulb Air Temp. (°C)
1	Office	77,854	36	9.0	35	30	60	10	24
2	Indoor Game Hall	26,400	2	2.5	50	0	50	8	25
3	School	142,169	7	4.0	20	0	50	10	24
4	Hospital	50,168	10	10.0	35	25	50	13	24
5	Terminal and Museum	44,400	6	4.0	50	30	50	5	24
6	Hotel	45,198	25	20.0	15	25	62.5	10	24
7	Retail	518,329	12	4.0	50	30	60	6.5	24
8	Mass Transit Station	2,400	2	3.0	50	30	0	5	25

day types (weekday, weekend and holiday) were assigned, making reference to the local building energy code published by the Hong Kong Special Administrative Region Government (HKSAR, 2003).

The model buildings were taken as rectangular blocks to represent typical built form. The building heights were governed by the maximum height limit of the development, and were assigned to reflect the appropriate functions. Two different types of building façade were adopted, namely the conventional tile cladding concrete wall and the curtain wall, with U-values of 1.9 and 0.4 W/(m^2K), respectively. The curtain wall was generally applied to the office buildings, hotels, retail, cruise terminal and museum. The window-to-wall ratio and the properties of glazing material (with shading coefficient varying from 0.32 to 0.45) were set based on the overall thermal transfer value (OTTV) estimation. This was governed by the local building regulation (HKSAR, 1995).

With a dynamic building simulation software EnergyPlus (DoE, 2005) and the hourly weather data of a typical meteorological year (TMY) developed for Hong Kong (see Chapter 3), the thermal load analysis of each generic model building was executed as shown in Figure 4.1. The hourly cooling loads of each typical model building of all the building categories were determined.

The hourly cooling load dataset of each category was normalised by the assigned gross floor area (GFA) of the corresponding typical building. Through this process, the normalised cooling load profiles of each category expressed in terms of cooling load intensity (*CLI*) in W/m^2 were determined. For the development site with *n* numbers of

building categories (n = 8 in the present study), the hourly district cooling load (DCL) profile and the DCL matrix (with elements in MW) can be determined as illustrated in Figure 4.2. A_1 to A_n in Figure 4.2 represent the GFA of each building category. DCL_{max} is the peak value sorted (i.e. the peak of the coincident load) from the DCL matrix elements and is the required cooling capacity of the DCS plant, i.e. the total installed capacity; while the year-round hourly cooling load of the entire district are DCL_1 to DCL_{8760} (Chan *et al.*, 2006b).

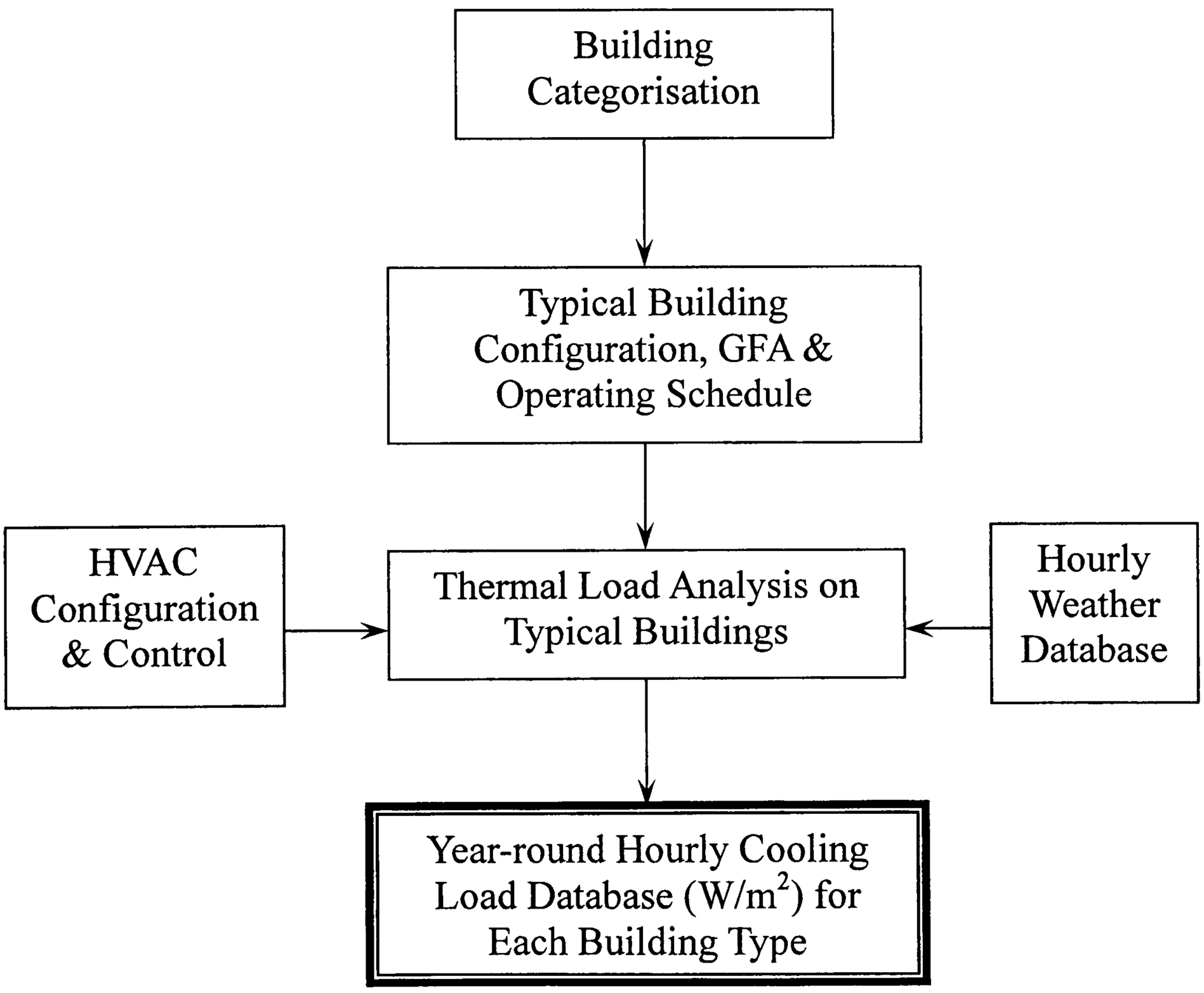


Figure 4.1 Flow chart for building modelling and cooling load prediction

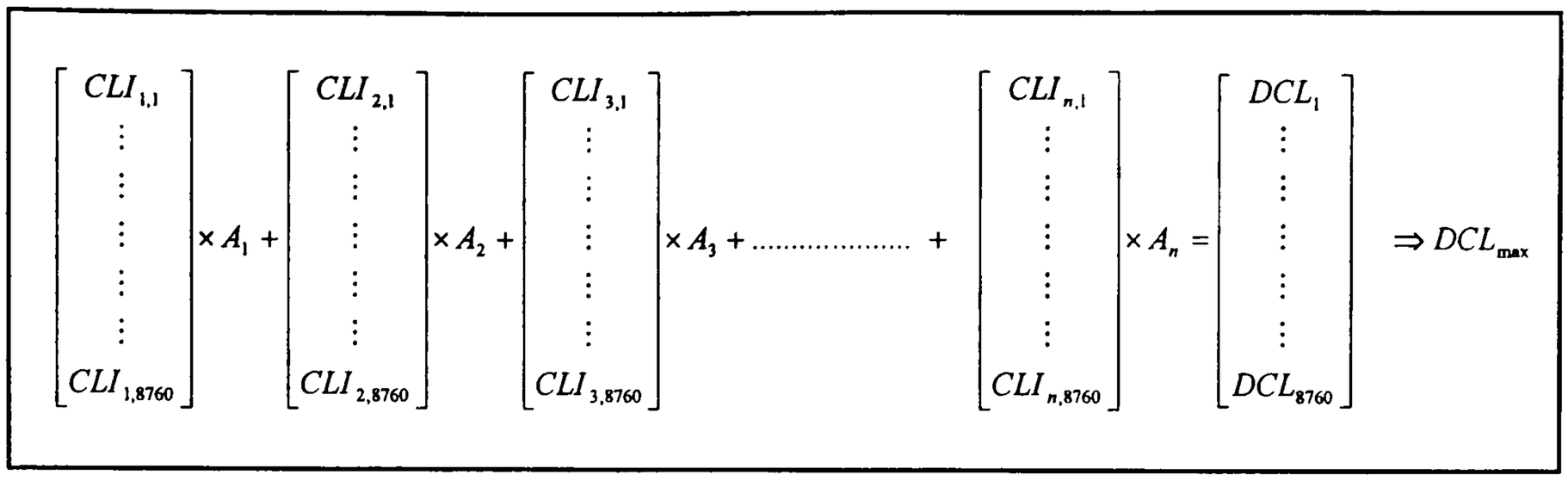


Figure 4.2 Determination of hourly and peak cooling loads in a district area

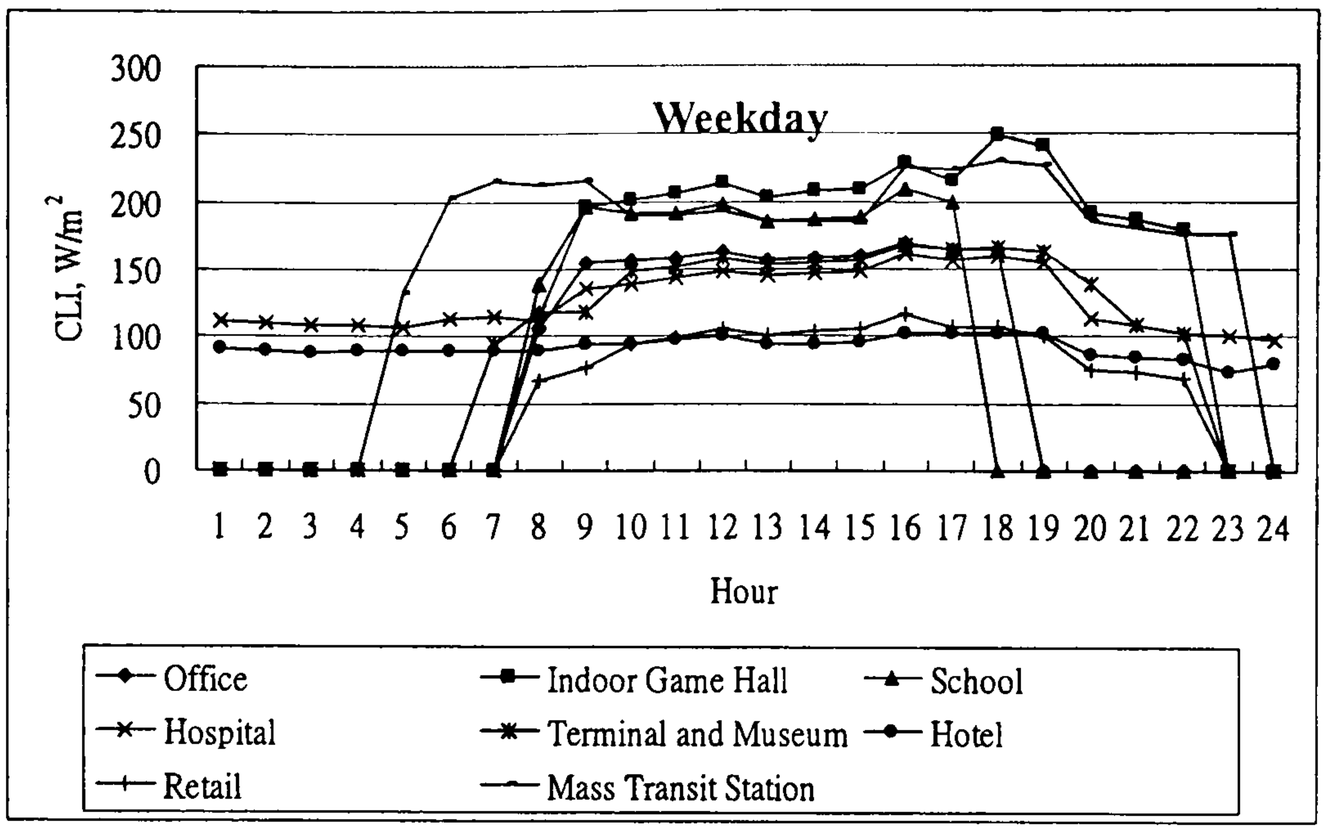
The peak cooling loads of each building category are listed in Table 4.2. They are independent loads which appear at different day types. For example, peak cooling load of office building category occurs in weekday while that of indoor game hall appears on Sunday. Across the eight building categories, retail is the top contributor to the cooling demand. Schools are the second major contributor, followed by the office category. It can also be seen that the cooling demand from retail dominates the total cooling demand as the GFA of this building category accounts for 57.2 % of the total GFA in the district. For determining the actual cooling loads which to be handled by the DCS, the instantaneous hourly cooling loads have to be determined. The corresponding coincident peak district cooling demand is 137.4 MW (DCL_{max}). This value, when being divided by the sum of independent peak loads of each building category (176.2 MW), gives a diversity factor of 0.78. The diversity factor is an indicator of one of the main advantages from a DCS over the chiller plants to be installed in each individual building. Since the peaks of the cooling loads of different building types occur at different time, the peak of the coincident district cooling load should be smaller than the sum of the peaks of individual buildings. This can help to avoid an oversized chiller plant because the total cooling load is not simply the summation of the individual and independent peak cooling loads. In addition, the

multiple chiller design for a centralised chiller plant can provide greater flexibility in optimising chiller plant operation. Energy saving, plant-cost and running-cost savings, plant size reduction, better plant maintenance and management are the subsequent benefits from the DCS.

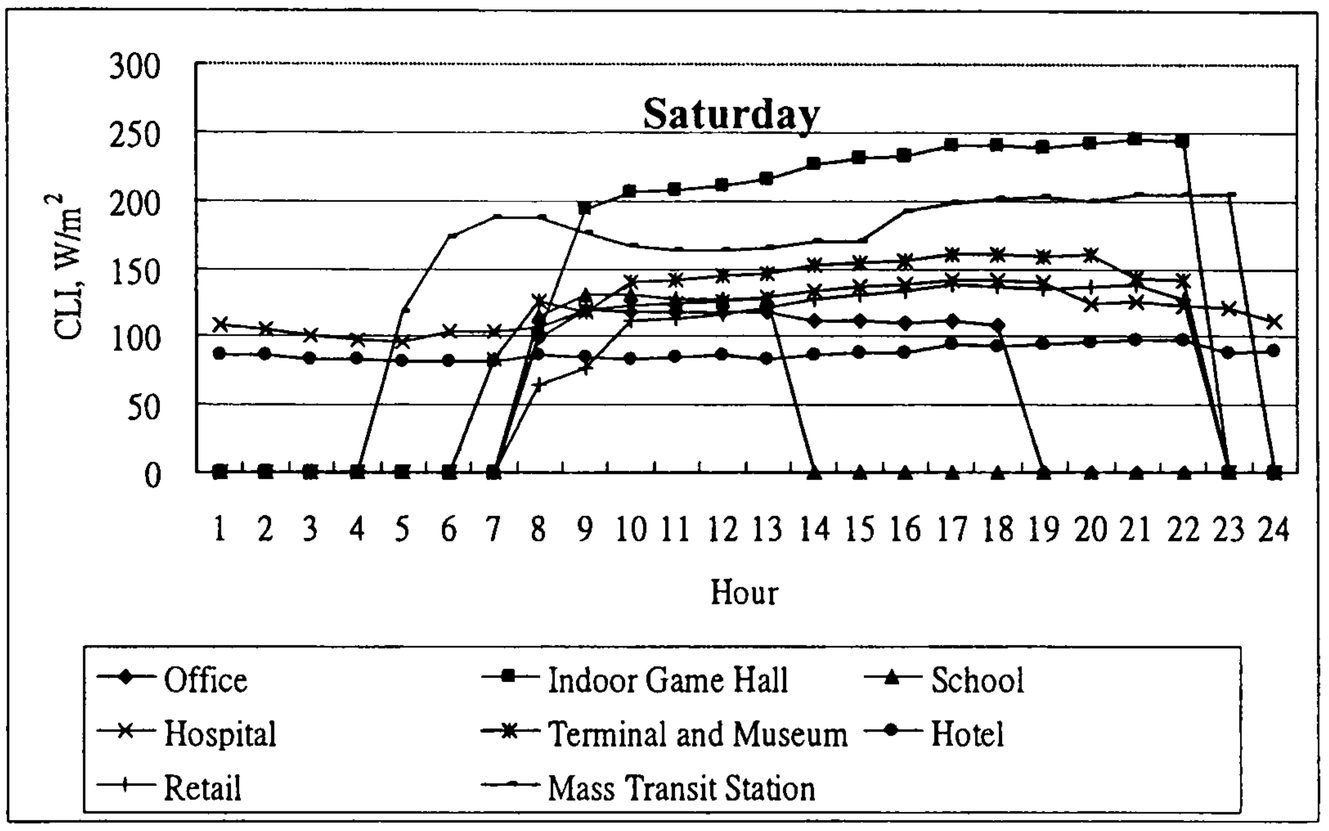
Table 4.2
List of peak cooling load data

Category No.	Description	Peak Load Day Type	Peak CLI (W/m^2)	Category Peak Cooling Load (MW)
1	Office	Weekday	170	13.2
2	Indoor game hall	Sunday	278	7.3
3	School	Weekday	211	30.0
4	Hospital	Weekday	162	8.1
5	Terminal and museum	Sunday	194	8.6
6	Hotel	Weekday	105	4.7
7	Retail	Sunday	200	103.7
8	Mass transit station	Weekday	230	0.6
		Sum of independent peak cooling loads =		176.2

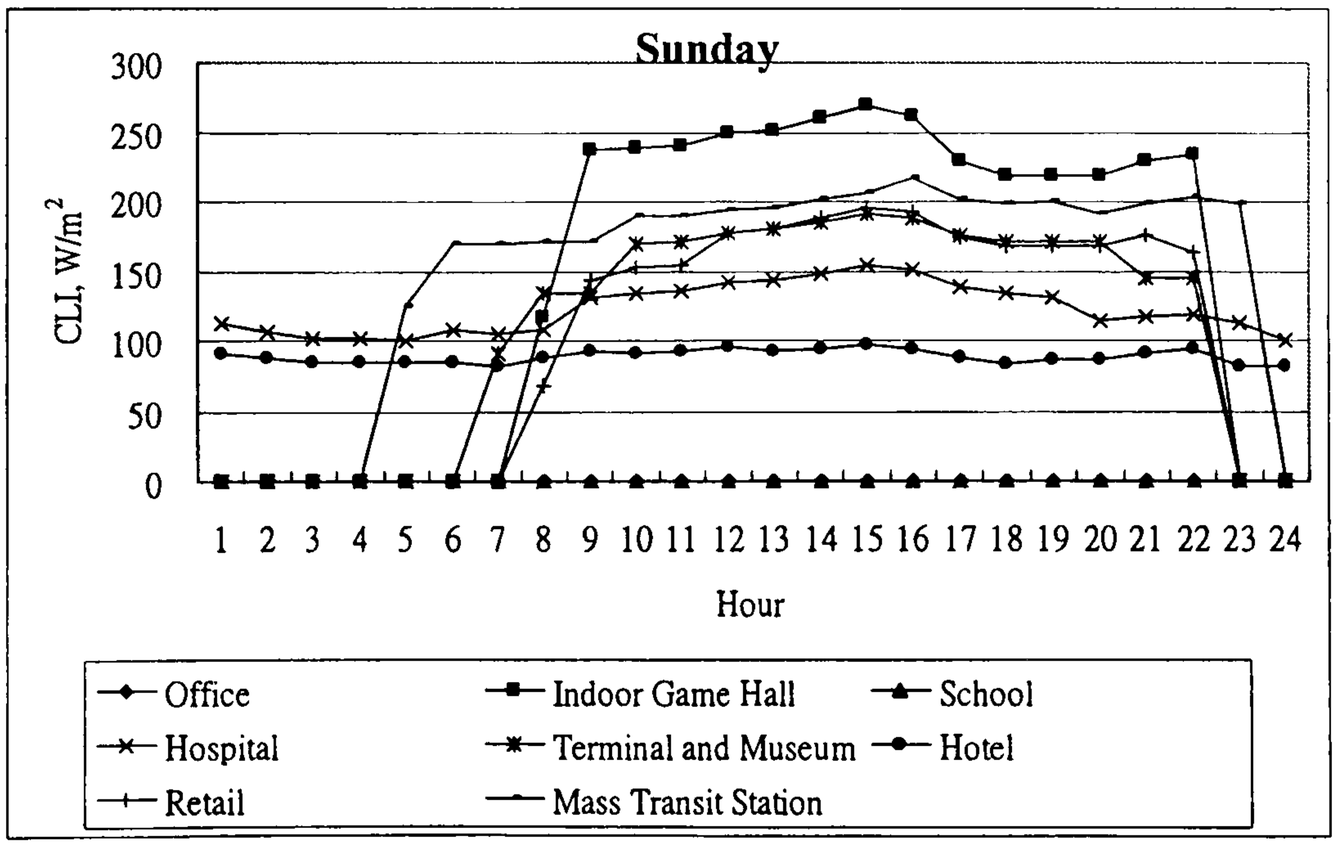
Figures 4.3(a) – (c) show the daily cooling load intensity profiles of the design month (July) for all the eight building categories, and for a weekday, Saturday and Sunday, respectively. Office and school buildings have very different cooling load patterns from the retail buildings. The former have a relatively high cooling load during the office hours but zero load on holidays. The peak load normally occurs during a weekday afternoon. For retail, however, there are longer daily operating hours and this cooling load pattern is relatively the same for all days with the peak occurs during a holiday afternoon. On the other hand, hospital and hotel buildings have space cooling demand round the clock everyday. In mass transit railway stations, there are high cooling demands up to midnight and the peak load occurs in the evening. Both the indoor game hall and the museum have the maximum cooling demand on Sunday as expected.



(a)



(b)



(c)

Figure 4.3 Daily variation of cooling load intensity for the eight building categories in design month

Figure 4.4 shows the design cooling load profiles of the entire district in July. Comparing the three day types, both weekday and Sunday profiles have the maximum load in the evening while the Saturday profile shows the maximum load in the early afternoon. Both Saturday and Sunday profiles extend their high cooling demands to nighttime. The year-round hourly cooling profile to be handled by the district cooling plant is shown in Figure 4.5. This result forms a data base of the space cooling loads which will be used for evaluation of hourly and annual pumping energy consumption of the distribution pumps in the piping network which will be presented in the next section.

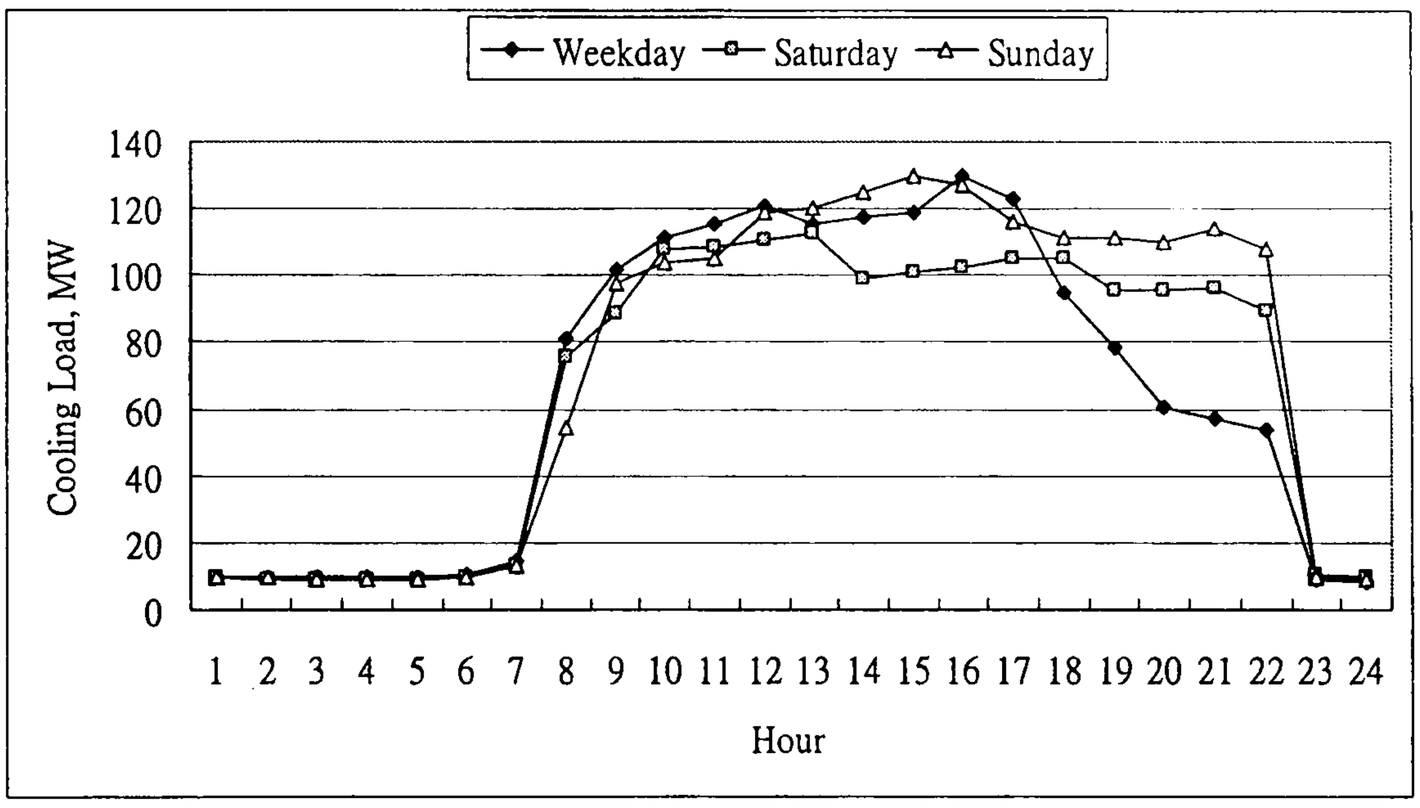


Figure 4.4 Cooling load profiles of district cooling system plant in design month

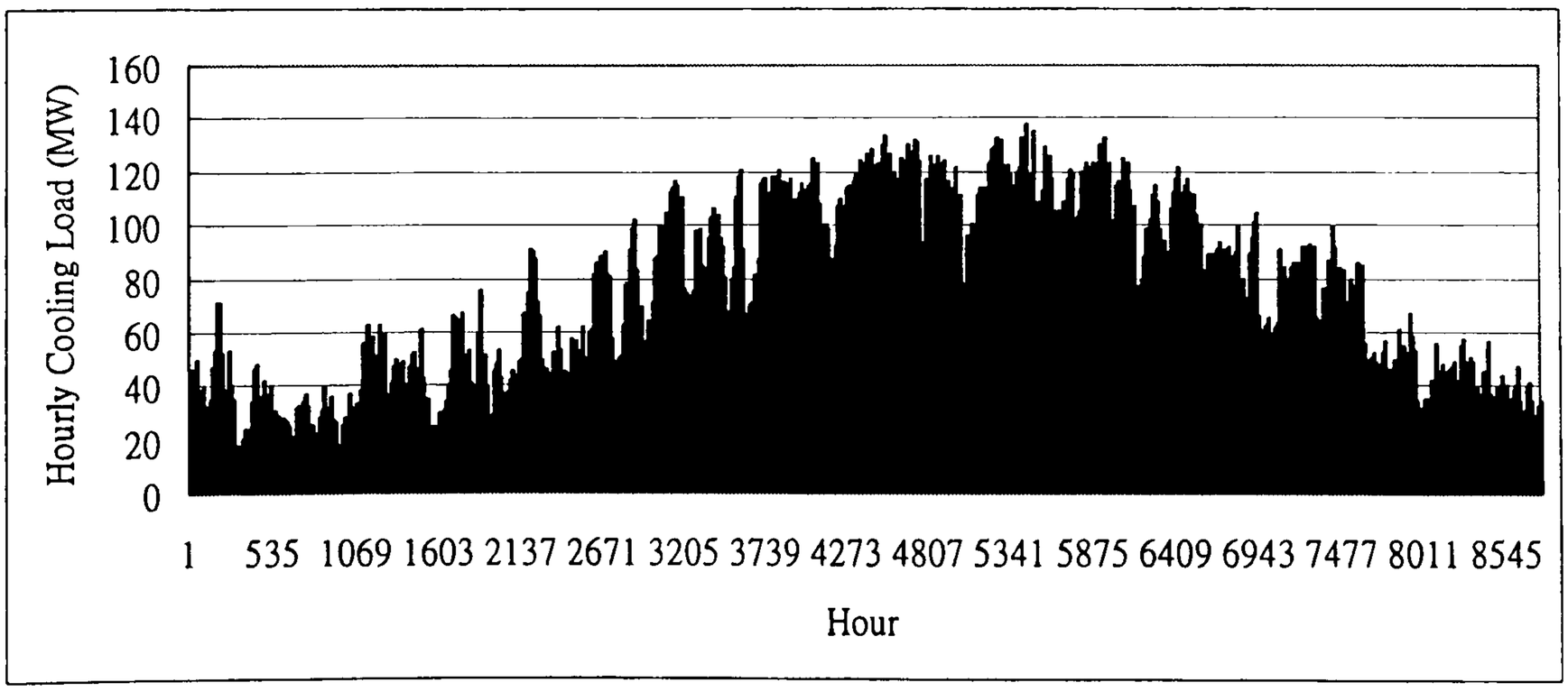


Figure 4.5 Cooling load profile of district cooling system plant in one year

4.3. Problem Formulation

In the hypothetical site of reclaimed land with different building mix including offices, hotels, retails, schools, hospitals and mass transit railway station, the locations of the consumer buildings (nodes) are pre-fixed. The piping network is modelled by a graph with undirected links. Either a radial or a tree-shaped network or a mix of both are allowed. The objective is to find the optimal/near-optimal piping configuration of a DCS that minimises the infrastructure (piping) cost compatible with the minimum pumping energy cost as listed in Equation (4.1) below.

$$\min \quad \left\{ Z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} C_{ij} x_{ij} + \alpha \sum_{k=1}^{8,760} Hourly\ Pumping\ Cost_k \right\} \tag{4.1}$$

subject to:

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_{ij} = n-1 \tag{4.2}$$

$$\sum_{\substack{i \in U \\ j>1}} \sum_{j \in U} x_{ij} \leq |U| - 1, \qquad U \subset V, \qquad |U| \geq 2 \tag{4.3}$$

where

n	=	number of nodes in the piping network
C_{ij}	=	amortised cost of (i, j) pipe segment over 30 years = $L_{ij} \times CPUL_{ij}$
L_{ij}	=	pipe length of (i, j) pipe segment
$CPUL_{ij}$	=	amortised cost per unit length of (i, j) pipe segment which depends on pipe diameter
x_{ij}	=	$\in \{0, 1\}$ is the decision variable $x_{ij} = 1$ if a pipe link (i, j) is connected $x_{ij} = 0$ if a pipe link (i, j) is not connected
α	=	weight factor (value of 6.73 was used in the present study)

Equation (4.2) is true for all feasible piping networks in which n nodes must have $n - 1$ links. Inequality (4.3) states that if there are more than $|U|$ - 1 links connecting the nodes of a subset U, this subset U will contain a cycle which is not allowed in this study.

Equation (4.1) shows that it is a multi-objective optimisation problem. In multi-objective optimisation problems, the objectives are usually conflicting so that further improvement with respect to one objective may affect the improvement reached on another objective. It is extremely rare that a single tuple optimises all objective functions simultaneously. Instead, there normally exist several tuples that provide compromise or trade-off with respect to the problem objectives.

For solving multi-objective optimisation problems, there are mainly two approaches. The first one is called generating approach or *a posteriori* method. This is a search-then-decide based strategy. It consists of finding as many of non-dominated solutions then selecting a solution from these. This strategy decouples the solutions from the decision making process. This technique needs the decision makers to make a necessary value judgment by selecting from the entire set of Pareto solutions.

The second one is a preference-based approach or *a priori* method. It is a decide-then-search strategy in which prior knowledge of the preference structure over all the objectives is needed. The decision maker has to encapsulate his/her preferences, expert knowledge and judgement about the components of the objective function. With the preference information, a compromised or preferred solution can then be identified. The most commonly applied method under this scheme is a weighted sum approach which combines the multi-objective functions into a single objective function.

This technique was used to yield a weight vector that mirrored the relative importance of each objective function.

$$\max \quad z(x) = \sum_{k=1}^{q} w_k f_k(x) \tag{4.4}$$

$$\text{s.t.} \quad x \in S$$

where

w_k	=	weight representing as the relative importance of an objective compared to other objectives
$f_k(x)$	=	objective function
S	=	feasible region in the decision space

The decision maker needs to assign weights to each of the objectives so that the weighted sum of the objectives can be transformed into a single objective function. Through an optimisation process, a single design solution which minimises the weighted sum of the objectives can then be found. If the optimisation process is repeated with different weights used for the objectives, different non-dominated solutions along a Pareto front will result. The choice of weights can be defined through the life-cycle cost of a project, which is defined by a weighted sum of the capital and operating costs. This is a common and practical approach used by the developers in assessing the different design options of building and engineering projects (Itoh, *et al.*, 2000; Mariano-Romero & Alcocer-Yamanaka, 2005; Fragiadakis, *et al.*, 2006). In the present study, this weighted sum approach was adopted.

The life-cycle cost of a distribution piping network in a DCS is the sum of the piping

cost and the total pumping cost over the life cycle of the network. This life-cycle cost then forms the combined single objective function. Alternatively, the combined objective function can be presented on an annual basis which was adopted in this study. In order to place both the piping and pumping costs on an annual basis, the capital cost of the infrastructure (piping) was amortised over 30 years (assumed life cycle of the pipe work). The annual pumping cost at the 30^{th} year was calculated at an average discount rate of 6.56% (HKSAR, 2006) which results in a factor of 6.73 (value of α in Equation (4.1)). If different weights are applied to the combined single objective function, different optimal solutions along the Pareto front will be obtained. However, this will not invalidate the development of the optimisation method/algorithm described in this study.

For calculating the pumping energy cost, the hourly chilled water demand of each DCS consumer building, as developed in Section 4.2, were used. Based on each piping network generated, the chilled water flow rate along each pipe segment was calculated. Then the diameter of each pipe segment was determined using the chilled water flow rate and a maximum flow velocity of 1.2 m/s. According to the Darcy formula shown in Equation (4.5), the pressure drop along each pipe segment could be calculated, and hence the critical path could be identified.

$$h_f = \frac{f\, l\, Q^2}{3.03\, d^5} \tag{4.5}$$

where

h_f = head loss

f = friction factor

l = pipe length
Q = chilled water flow rate
d = pipe diameter

With the hourly total chilled water flow rate in the network and the pressure head along the critical path, the hourly pumping energy could be determined. The pumping energy was then used to calculate the corresponding pumping cost, which consisted of two components (energy charge and demand charge), making reference to the tariff structure of a local power generating company China Light & Power Limited in Hong Kong Special Administrative Region (CLP, 2006).

Under this structure, the tariff is the aggregate of the energy charge and demand charge. Table 4.3 lists the details of the energy charge while the data of demand charge are shown in Table 4.4. Based on the actual energy consumption of the distribution pumps, the energy charge is calculated. There are different rates offered for on-peak and off-peak periods. The basic tariff structure is established with a standard fuel cost. Through the Fuel Clause Account mechanism between the Government and the power company, the difference between the standard fuel cost and the actual fuel cost will be returned to the consumers by means of a rebate (i.e. the scheme of control rebate as shown in Table 4.3).

For the demand charge, it is calculated with the monthly maximum electricity demand in kVA. "kVA" is a measure of the actual demand for power that an installation puts on the electrical system. If the installation is inefficient or the load is unevenly distributed, the value of kVA will be increased. Through this part of tariff structure, the power company encourages the consumers to spread their electricity consumption

over a longer period, if possible, rather than concentrate the consumption in a short period of time. Therefore the power company can benefit from reducing the investment in electric power generating plant. The variation of cooling loads in a DCS is due to the differences in occupied periods of the different building categories, fabric and internal heat loads. This fluctuating pattern of the cooling demand determines the monthly maximum electricity demand as well as the monthly demand charge.

Table 4.3
Tariff structure of energy charge

Total Monthly Consumption	**Rate (Cents/Unit)**	**Scheme of Control Rebate (Cents/Unit)**	**Net Energy Charge Rate (Cents/Unit)**
On-Peak Period			
Each of the first 200,000 units	70.0	0.8	69.2
Each unit over 200,000	68.5	0.8	67.7
Off-Peak Period			
Each unit	62.5	0.8	61.7

"Unit" in this rate table means one kilowatt-hour (kWh) of electricity.
One HK$ equals to 100 cents.
"Off-peak Period" is the daily period between 21:00 hours and 09:00 hours and all Sundays and Public Holidays.
"On-peak Period" comprises all other hours.

Table 4.4
Tariff structure of demand charge

Based on the monthly maximum demand in kVA	
On-Peak Period	
Each of the first 650 kVA	HK$66.5
Each kVA over 650	HK$63.5
Off-Peak Period	
Each off-peak kVA up to the on-peak billing demand	HK$0.0
Each off-peak kVA in excess of the on-peak billing demand	HK$26.0

The cost of each pipe segment was determined according to their pipe lengths and diameters. The unit prices of the commercially available pipes were referred to the information published (Dandy, *et. al.*, 1996; Nedjah & Mourelle, 2005) and private communication with local suppliers. Then the sum of the pumping energy cost and the pipe work cost is equal to the total cost of the piping network. This total cost forms the central core of the objective function that to be optimised. The details of the optimisation methodology will be discussed in the next chapter.

CHAPTER 5
OPTIMISATION METHOD

As described in Chapter 2, applying an exhaustive search method for optimal design of distribution piping network in a district cooling system is computationally expensive and infeasible. After a detailed review, it was decided to use a genetic algorithm for studying the optimisation problem of the piping configuration. For a successful and efficient implementation of the genetic algorithm in the optimisation process, appropriate representation (encoding) of the piping configuration is one of the crucial factors. In this chapter, a detailed description of the optimisation methodology for the optimal piping configuration in this study will be presented. After that, various types of representation methods used in a genetic algorithm will be reviewed and an appropriate one will be selected for the present study. Finally, the advantages and approach of incorporating local search technique into the genetic algorithm will be discussed.

5.1 Optimisation Methodology

A flow chart of the optimisation methodology is shown in Figure 5.1. The optimisations are run under MATLAB (release 14) environment in the present study. As seen from the flow chart, an initial population of size m is generated randomly at the beginning of the optimisation process. An appropriate encoding method is used to represent the piping configurations in the population (the details of the encoding methods will be addressed in Section 5.2). This initial population is then decoded to form the piping configuration. After that, the function value is determined. As mentioned in Chapter 4 (Section 4.3), the function value of this problem is the total cost including the infrastructure (piping) cost and the pumping energy cost.

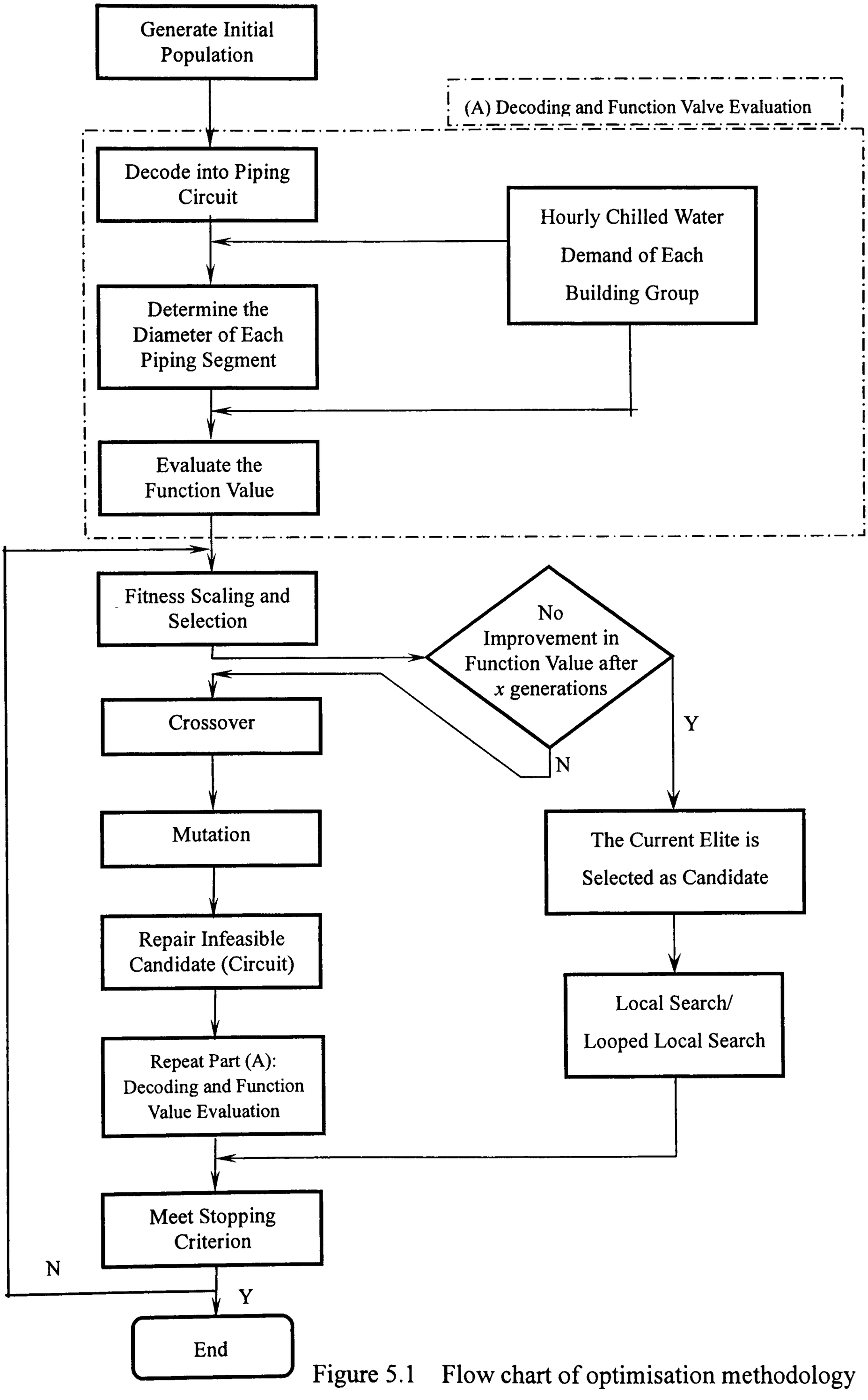


Figure 5.1 Flow chart of optimisation methodology

Firstly, based on the piping circuit formed and with the hourly chilled water demand of each building group (node) predicted, the chilled water flow rate at a particular hour along each pipe segment is determined. The process repeats for 8,760 times so that the peak hourly chilled water flow at each piping segment can be identified. Then the diameter of each pipe segment is determined using the corresponding peak chilled water flow rate and a maximum flow velocity of 1.2 m/s. The piping cost can then be calculated with the lengths, diameters and unit costs of each piping segment connecting the nodes. For the annual pumping energy cost, the pressure drop along each pipe segment is calculated, and hence the critical path can be identified. With the hourly total chilled water flow rate in the network and the pressure head along the critical path, the hourly pumping energy can be determined. The process repeats for all the hours in a month so that the monthly electricity consumption can be calculated. With the tariff information provided by a local power generating company, the monthly electricity cost (including energy cost and demand cost) can be determined. In a similar way, the annual pumping energy cost of a particular piping circuit can be obtained.

After the evaluation of the function value of each solution in the population, some of the solutions are selected as parents for reproduction of offspring. The solutions could be selected using a proportional selection operator, i.e. the expected number of copies a solution receives is assigned proportionally to its fitness. However, there may be a loss of genetic diversity and causes the algorithm to prematurely converge to suboptimal solution or collapse onto a false solution. This case will occur when the population contains a solution having exceptionally better fitness than the rest of the solutions in the population. On the other hand, if most of the population members have more or less the same or close fitness, every solution becomes equally likely to be selected.

This has the effect of a random search. In order to avoid the inherent difficulties caused by the proportional selection, scaling scheme is commonly used. In this study, ranking scaling is adopted. The approach is similar to the proportional selection operator except that the solutions are ranked according to the order of their fitness values. The rank of the least fit is defined to be 1 and the rank of the mostly fitted one is defined to be *m*, given a population of size *m*. Copies are allocated with the resulting selection probabilities of the solutions calculated using the ranked fitness values. It preserves a constant selection pressure by sorting the population on the basis of fitness, and then allocating selection probabilities to individuals according to their ranks, rather than their actual fitness values.

The parents are chosen for the next generation based on their ranked fitness values. Roulette wheel sampling is one of the commonly used sampling algorithms in which each slice has a width proportional in size to the ranked fitness and the sampling can be envisioned as spinning the roulette wheel. Despite the inherent simplicity of roulette wheel algorithm, it has been recognised that the roulette wheel algorithm does not in fact give a particularly good sample of the required distribution. In a generational algorithm, the entire population is replaced during each generation, so the probability distribution is sampled *m* times. This could be implemented by *m* independent calls to the roulette wheel procedure. But such an implementation may exhibit a high variance in the number of offspring assigned to each individual. It is possible that the individual with the largest selection probability may be assigned no offspring in a particular generation (Bäck, *et al.*, 2000). Whenever more than one sample is to be drawn from the distribution, the use of the stochastic universal sampling algorithm is preferred (Eiben & Smith, 2003).

The principle of the stochastic universal sampling (Baker, 1987) is that given a roulette wheel with slots whose sizes are proportion to the individual's selection probability and starting at a random position less than one full step, *m* steps of equal size are landed in so that the solutions are sampled by choosing them at evenly spaced intervals. With this sampling algorithm, it exhibits less variance than repeated calls to the roulette wheel algorithm. In the present study, this stochastic universal selection function is adopted.

The new generation is produced by three operators: elitism, crossover and mutation. For elitism, it retains a certain number of individuals which are guaranteed to survive in the next generation. In this study, elite count is set to 1, i.e., the best solution in the current generation is carried forward to the next generation.

Crossover is a process in which a new individual solution is created from the information contained within two or more parent solutions. It is considered as one of the most important features in genetic algorithms. Crossover worked by randomly mixing parts of the parents. *n*-point crossover has an inherent bias in that it tends to keep together genes that are located close to each other in the representation. Further, when *n* is odd, e.g. one-point crossover, there is a strong bias against keeping together combinations of genes that are located at opposite ends of the representation. These effects are known as positional bias. In contrast, uniform crossover does not exhibit any positional bias (Eiben & Smith 2003). In the present study, uniform (scattered) crossover operator is used. In uniform crossover, each gene has an equal chance of coming from either parent. The first child is generated in a way that each gene is randomly assigned to be that of either one or the other parent, with 50% probability.

The second child would then take the inverse mapping of the first child. One advantage of uniform crossover is that the genes inherited are independent of position.

The application of uniform crossover operator in the present study is illustrated by an example as shown below. The DCS distribution piping network is encoded with an Integer String method (details can be found in Section 5.2). Figures 5.2 and 5.3 show two example piping networks with integer-strings of {1, 34, 8, 5, 35, 23, 27, 9} and {13, 20, 3, 19, 6, 8, 22, 23}, respectively. The corresponding links of the integer strings can be found in Table 5.1.

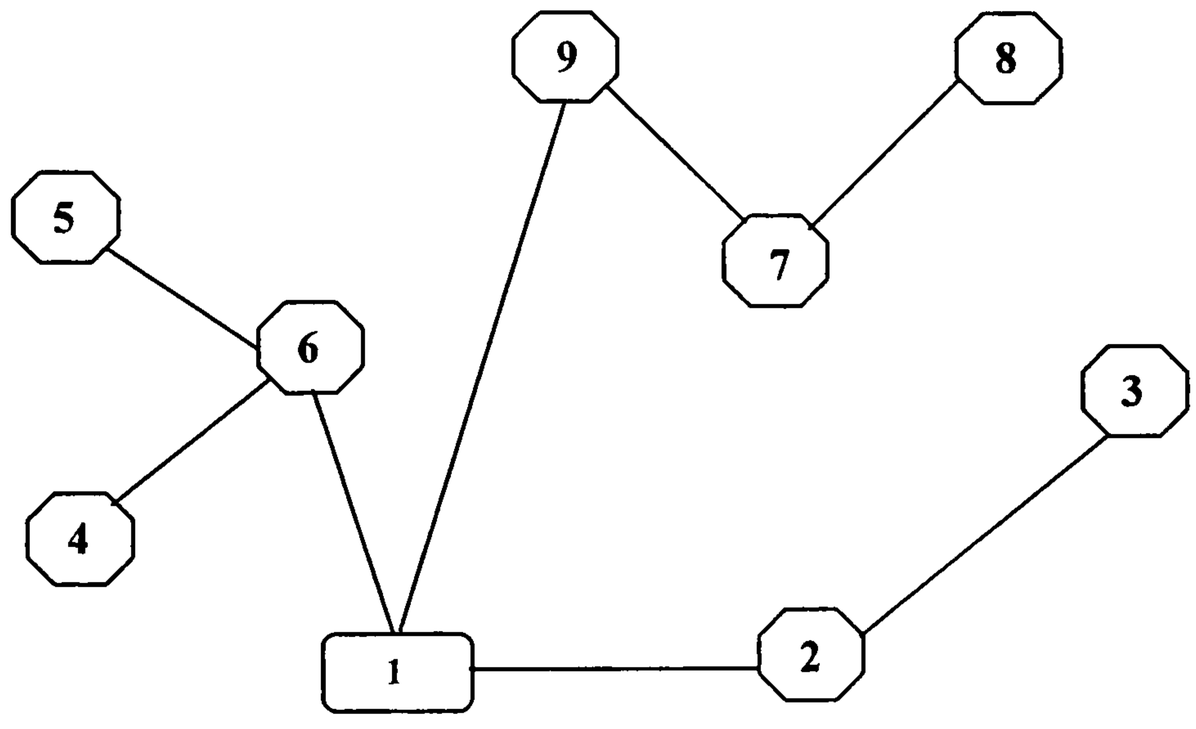


Figure 5.2 An example of piping network no.1 with integer-string {1, 34, 8, 5, 35, 23, 27, 9}

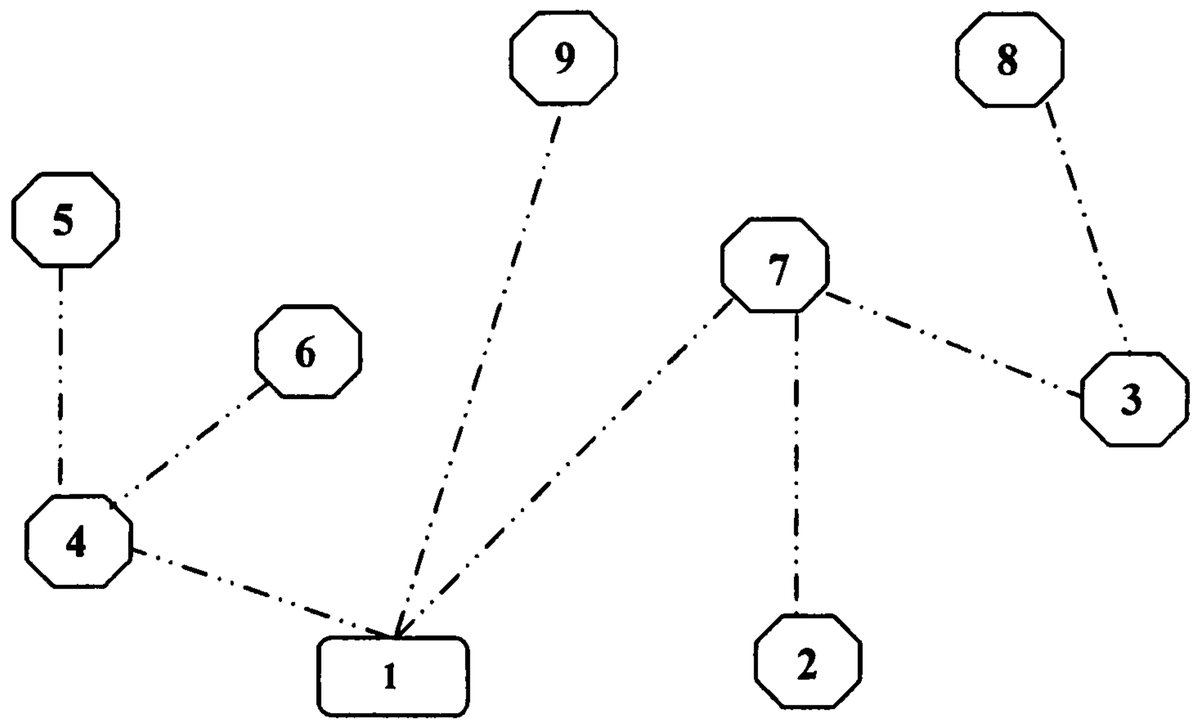


Figure 5.3 An example of piping network no.2 with integer-string {13, 20, 3, 19, 6, 8, 22, 23}

Table 5.1
Integer string encoding for piping network

Integer label	Link	Integer label	Link	Integer label	Link
1	1-2	13	2-7	25	4-8
2	1-3	14	2-8	26	4-9
3	1-4	15	2-9	27	5-6
4	1-5	16	3-4	28	5-7
5	1-6	17	3-5	29	5-8
6	1-7	18	3-6	30	5-9
7	1-8	19	3-7	31	6-7
8	1-9	20	3-8	32	6-8
9	2-3	21	3-9	33	6-9
10	2-4	22	4-5	34	7-8
11	2-5	23	4-6	35	7-9
12	2-6	24	4-7	36	8-9

Uniform crossover is implemented by generating a string of L random variables from a uniform distribution over [0,1]. In each position, if the value is below a parameter p (usually 0.5), the gene is inherited from the first parent, otherwise from the second parent. The second offspring is created using the inverse mapping. As seen from Table 5.2, through uniform crossover, the 2nd, 5th and 7th genes are chosen and offspring no.1 is formed as $\{P_{1,1}, P_{2,2}, P_{1,3}, P_{1,4}, P_{2,5}, P_{1,6}, P_{2,7}, P_{1,8}\}$ and that for offspring no.2 is $\{P_{2,1}, P_{1,2}, P_{2,3}, P_{2,4}, P_{1,5}, P_{2,6}, P_{1,7}, P_{2,8}\}$. The two corresponding offspring networks are shown in Figures 5.4 and 5.5, respectively.

Table 5.2
Uniform crossover for the example piping networks

Parent 1	$P_{1,1}$	$P_{1,2}$	$P_{1,3}$	$P_{1,4}$	$P_{1,5}$	$P_{1,6}$	$P_{1,7}$	$P_{1,8}$
	1	34	8	5	35	23	27	9
Parent 2	$P_{2,1}$	$P_{2,2}$	$P_{2,3}$	$P_{2,4}$	$P_{2,5}$	$P_{2,6}$	$P_{2,7}$	$P_{2,8}$
	13	20	3	19	6	8	22	23
Random variables*	0.34	0.65	0.28	0.24	0.85	0.39	0.77	0.31
Offspring 1	1	20	8	5	6	23	22	9
Offspring 2	13	34	3	19	35	8	27	23

* Random variables drawn uniformly from [0,1] was used to determine the inheritance. This consists of using a pseudorandom number generator to generate a series of values from a given probability distribution.

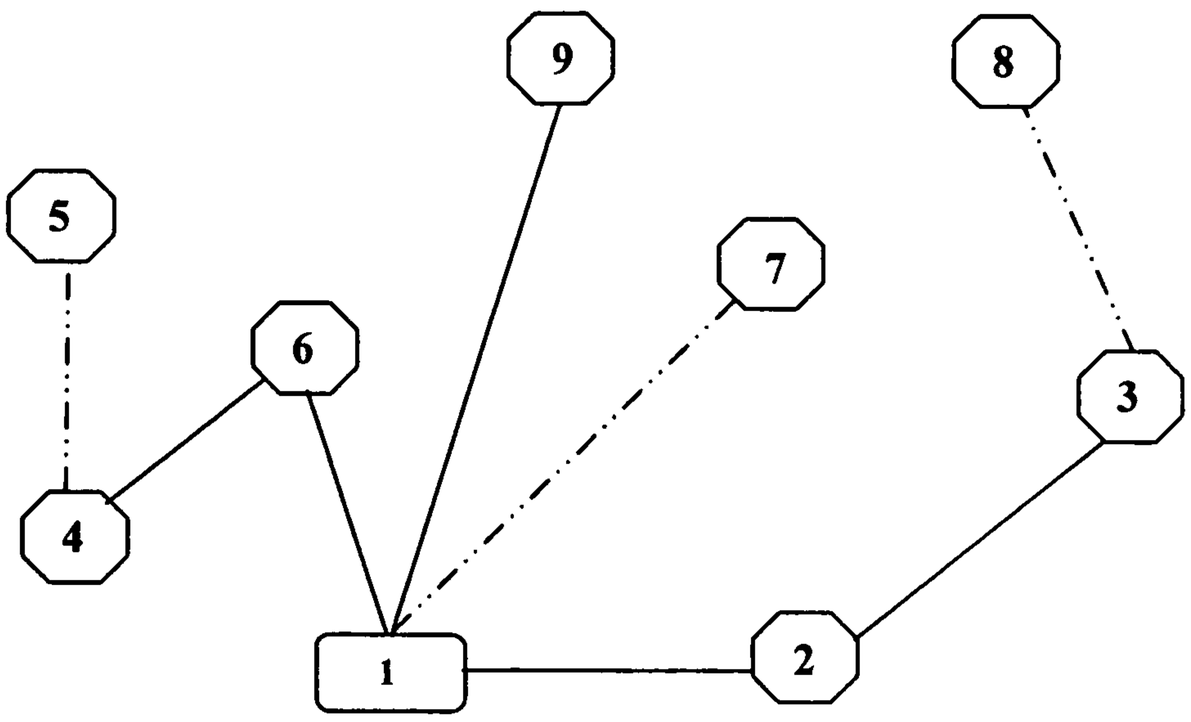


Figure 5.4 Offspring network no.1 with integer-string {1, 20, 8, 5, 6, 23, 22, 9}

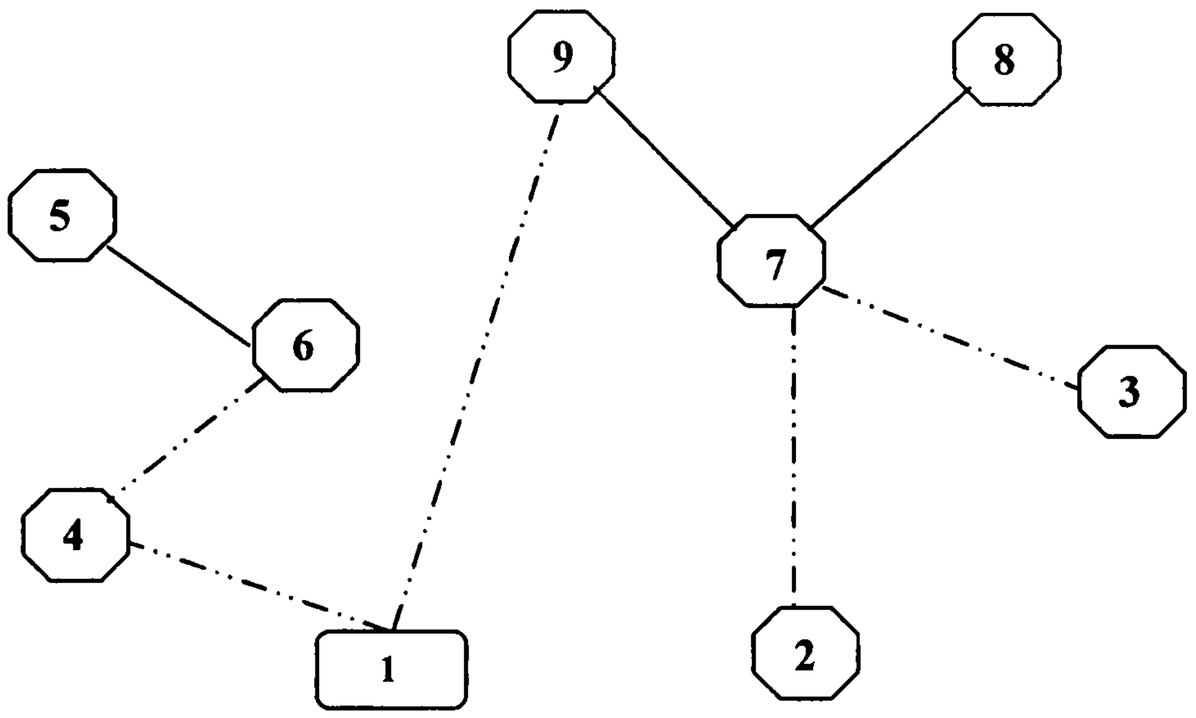


Figure 5.5 Offspring network no.2 with integer-string {13, 34, 3, 19, 35, 8, 27, 23}

Mutation can be used as a background operator, supporting the crossover operator by assuring that the full range of the gene values is accessible to the search. A mutation operator is always stochastic, i.e. the output (offspring) depends on the outcomes of a series of random choices. In genetic algorithm, it has traditionally been seen as a

background operator to fill the gene pool with "fresh blood". Mutation is described as exploitative and it can optimise present information within an already discovered promising region; and create small random deviations and thereby not wandering far from the parents. Crossover and mutation complement each other since only crossover can bring together information from both parents. On the other hand, only mutation can introduce completely new information. Mutation generally refers to the creation of a new solution from one parent. The mutation function makes small random changes in the individuals of the population which provides genetic diversity and a broader search space for GA. In this study, uniform mutation is adopted.

In uniform mutation, it is to change the value of a selected gene (according to the mutation probability) within its domain given by a lower bound, L_i and upper bound, U_i. The values of the genes are drawn uniformly randomly from $[L_i, U_i]$. The result of such mutation is a random value from the domain $[L_i, U_i]$.

The working principle of uniform mutation can be illustrated by an example as follows. A sample of DCS distribution network is shown in Table 5.3 and Figure 5.6 with an integer string of {1, 34, 8, 5, 35, 23, 27, 9}.

Table 5.3
Integer string encoding for an example piping network

Integer label	Link	Integer label	Link	Integer label	Link
1	1-2	13	2-7	25	4-8
2	1-3	14	2-8	26	4-9
3	1-4	15	2-9	27	5-6
4	1-5	16	3-4	28	5-7
5	1-6	17	3-5	29	5-8
6	1-7	18	3-6	30	5-9
7	1-8	19	3-7	31	6-7
8	1-9	20	3-8	32	6-8
9	2-3	21	3-9	33	6-9
10	2-4	22	4-5	34	7-8
11	2-5	23	4-6	35	7-9
12	2-6	24	4-7	36	8-9

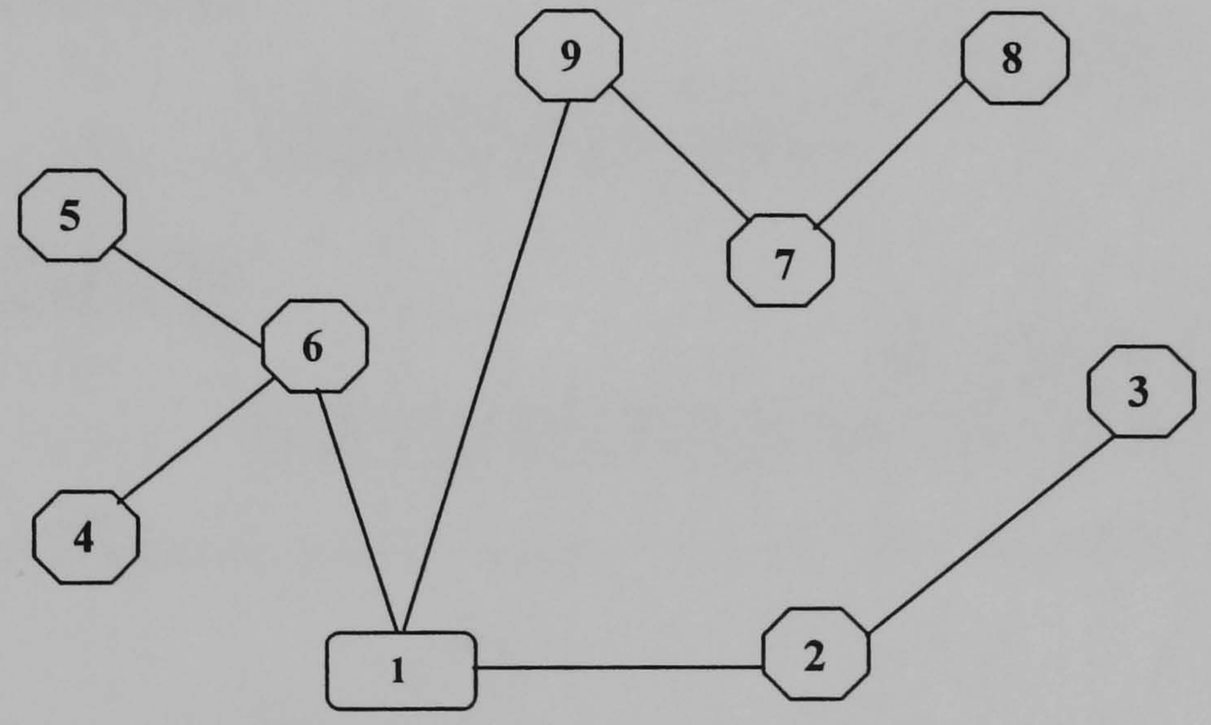


Figure 5.6 An example of piping network with integer-string {1, 34, 8, 5, 35, 23, 27, 9}

The 3rd and 7th genes are selected randomly. Their values are replaced by two random numbers selected uniformly from the range [1,36]. In this example, random numbers 19 and 30 are assumed to be generated. As shown below, the value of 8 is replaced by

19; and the other value of 27 is replaced by 30.

$$\langle 1, 34, \textcircled{8}, 5, 35, 23, \boxed{27}, 9 \rangle \rightarrow \langle 1, 34, \textcircled{19}, 5, 35, 23, \boxed{30}, 9 \rangle$$

The corresponding changes are shown in Table 5.4 and Figure 5.7; and the new piping network is illustrated in Figure 5.8.

Table 5.4
Integer string encoding for an example piping network after uniform mutation

Integer label	Link	Integer label	Link	Integer label	Link
1	1-2	13	2-7	25	4-8
2	1-3	14	2-8	26	4-9
3	1-4	15	2-9	27	5-6
4	1-5	16	3-4	28	5-7
5	1-6	17	3-5	29	5-8
6	1-7	18	3-6	30	5-9
7	1-8	19	3-7	31	6-7
8	1-9	20	3-8	32	6-8
9	2-3	21	3-9	33	6-9
10	2-4	22	4-5	34	7-8
11	2-5	23	4-6	35	7-9
12	2-6	24	4-7	36	8-9

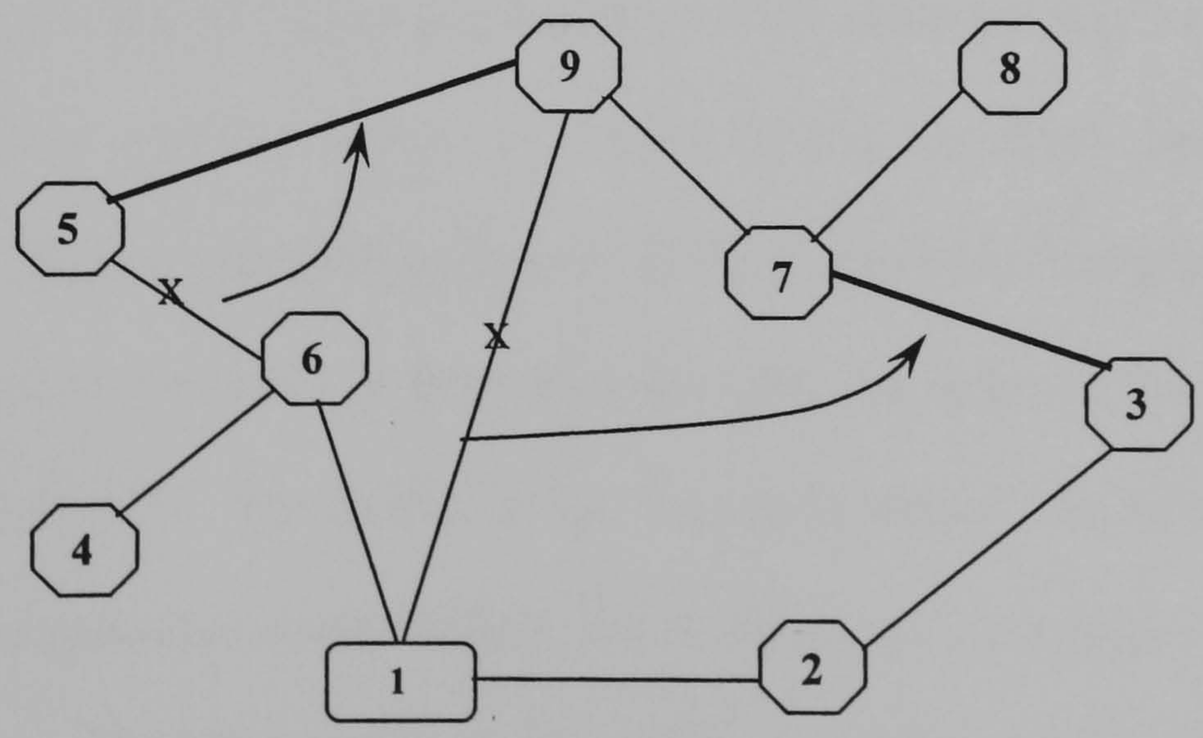


Figure 5.7 An example piping network with 2 links deleted and 2 new links added

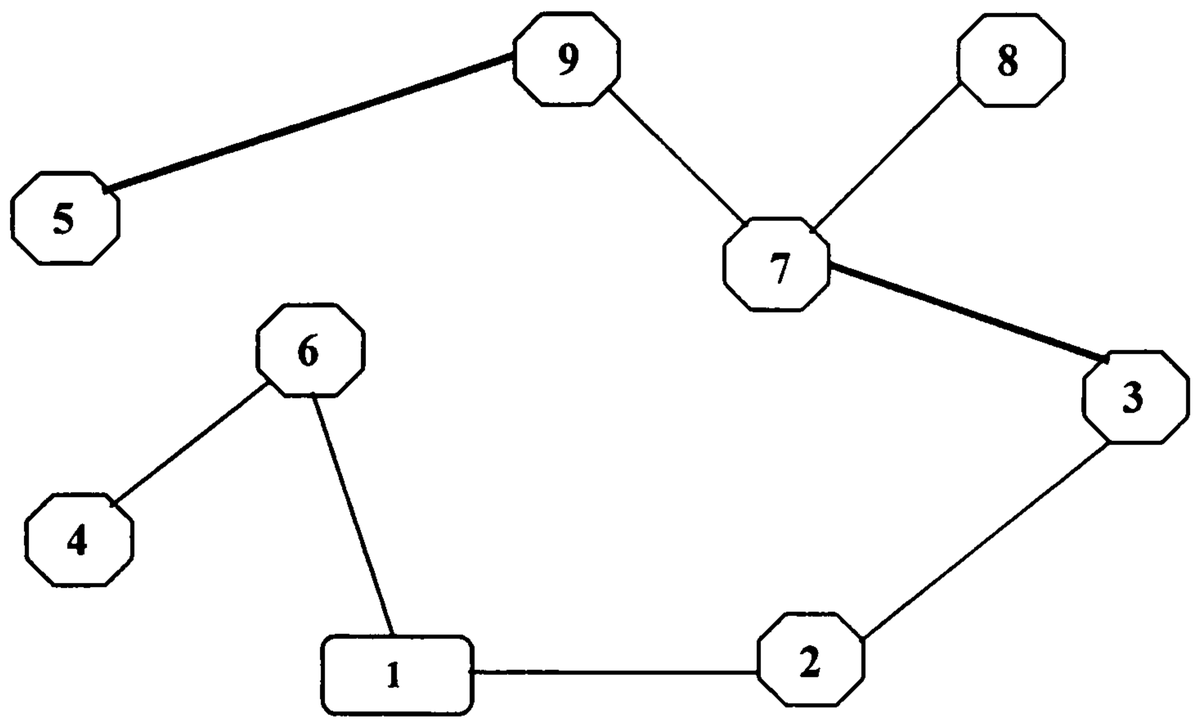


Figure 5.8 An example of piping network after mutation with integer-string {1, 34, 19, 5, 35, 23, 30, 9}

Infeasible offspring may be generated by the crossover and/or mutation operators. Therefore, a subroutine "repairalgorithm" has been developed to check the feasibility of the candidates. If it is an infeasible candidate such as a case with non-connected node(s); with repeated link and/or with looped circuit, it should be detected and then repaired by either being assigned a penalty value or to be repaired until a feasible circuit can be formed. Killing an infeasible candidate is not recommended since a good solution may result by breeding from feasible and infeasible candidates. For repairing an infeasible circuit, the disconnected node(s) or disconnected tree circuit(s) is firstly searched. Then repeated link(s) or looped link(s) are identified. The link will be connected to the disconnected node(s) or tree circuit(s) and the process repeated until a feasible circuit is formed. A flowchart showing the steps of the "repairalgorithm" is shown in Figure 5.9. The details of the "repairalgorithm" can be found in Section 6.3. Once a new population is established, the decoding of a candidate into a piping circuit and evaluation of function value repeat until a preset stopping criterion is met.

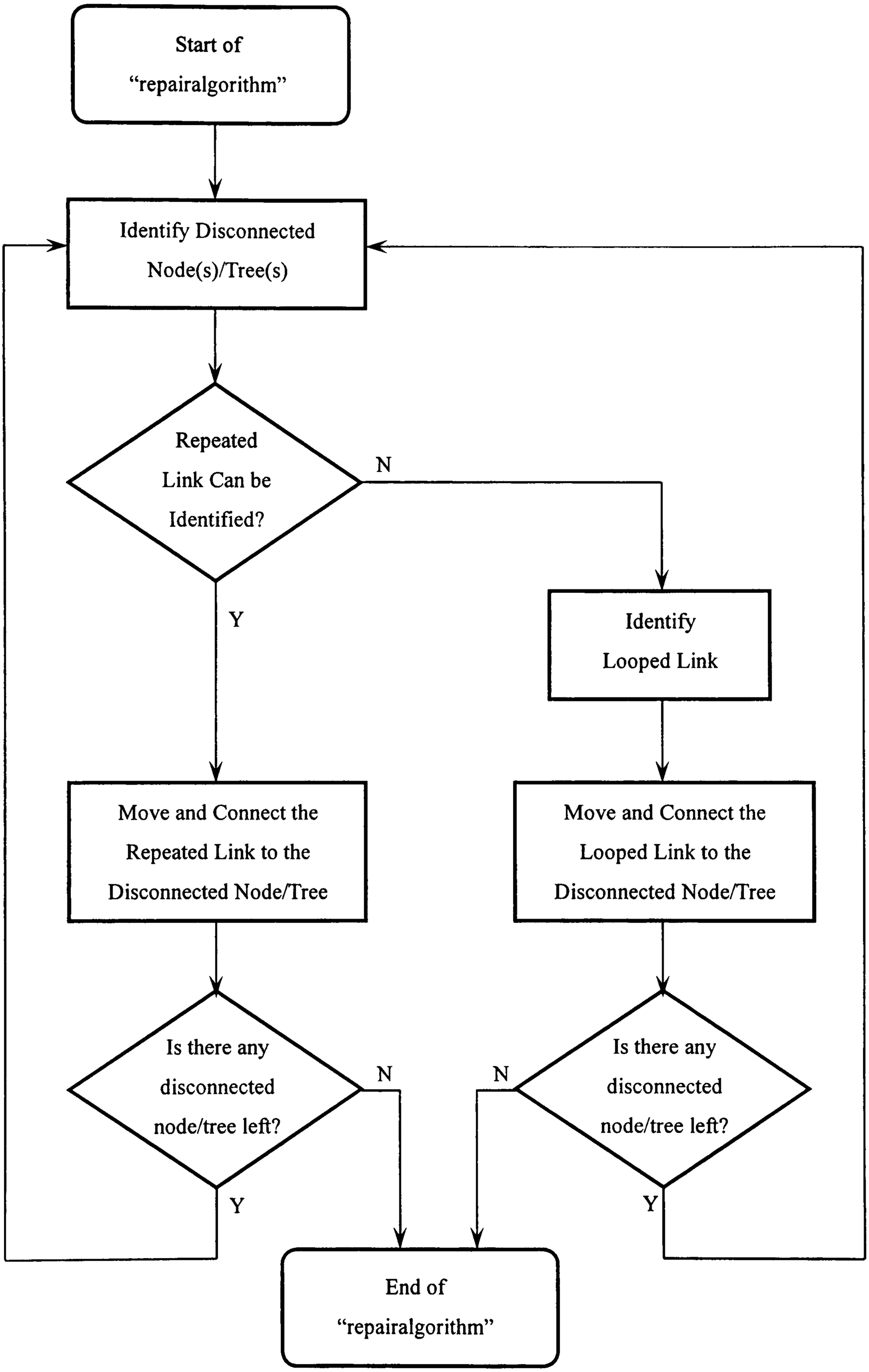


Figure 5.9 Flow chart of "repairalgorithm"

The GA performs a global sampling of the search domain. This type of heuristic algorithm for solving optimisation problem can find near-optimal solutions within a reasonable amount of computational time. However, it does not refine the solution efficiently. A local search algorithm incorporated into the GA can search for a better solution, generally starting with a feasible solution found by the crossover/mutation operators. At each iteration, an improving solution may be found by searching the neighborhood of the current solution. In the present study, a candidate (the elite) is selected and input into a subroutine "Local Search/Looped Local Search" for searching a better solution. The details of the Local Search/Looped Local Search will be presented in Section 5.3

5.2 Encoding the Piping Configuration for Genetic Algorithm

In a piping network, nodes are connected by links so that a configuration in a format of radial, tree, or a mix of them can be formed. For the optimisation of piping network using a genetic algorithm, the piping configuration has to be encoded into a string of numbers which is treated by the genetic algorithm as a candidate to undergo various processes such as mutation and crossover. It is crucial to have an encoding method which is able to represent all possible piping configurations otherwise it could happen that the optimal solution can never be reached because it cannot be encoded. In the field of evolutionary algorithms, there are five commonly used encoding methods for representing piping circuits, namely Prüfer Number, Link and Node Biased (LNB), Characteristic Vector (CV), Network Random Keys (NetKeys) and Direct Tree Representation (NetDir) (Rothlauf, 2002). The details of the encoding and decoding procedures of these encoding methods with illustrative examples are listed in Appendix II for reference.

Before going into the features of the various encoding methods, the following paragraphs list the details of three terms which are commonly used to describe the characteristics of encoding method. (Rothlauf, 2002).

(i) Redundancy

In encoding methods, redundancy is defined as that on average one phenotype is represented by more than one genotype. The performance of genetic algorithms can be influenced by the use of a non-uniformly redundant encoding method. If the optimal solution is over-represented by the encoding method used, the performance of genetic algorithms is increased and vice versa. Uniform redundancy, however, has no influence on the performance of genetic algorithms.

(ii) Distance Distortion/Locality

The locality of a representation describes how well the neighbouring genotypes correspond to the neighbouring phenotypes. The locality of representation is high if all the neighbouring genotypes correspond to the neighbouring phenotypes. In contrast, if some neighbouring genotypes do not correspond to neighbouring phenotypes, the locality will be low. The locality d_m of a representation method can be defined as:

$$d_m = \sum_{d^g_{x_i,x_j}=d^g_{\min}} \left| d^p_{x_i,x_j} - d^p_{\min} \right| \qquad (5.1)$$

where

$d^p_{x_i,x_j}$ = phenotypic distance between the phenotypes x^p_{xi} and x^p_{xj}

$d^p_{\min}$ = minimum distance between two neighbouring phenotypes

$d^g_{x_i,x_j}$ = genotypic distance between the corresponding genotypes x^g_{xi} and x^g_{xj}

$d^g_{\min}$ = minimum distance between two neighbouring genotypes

Without loss of generality, it is assumed that $d^g_{\min} = d^p_{\min}$. If d_m is equal to zero, the genotypic neighbours correspond to the phenotypic neighbours, the encoding has perfect locality, and the complexity of the optimisation problem remains unmodified. Low locality of a representation (high d_m values) would result in high distance distortion and vice versa. Genetic algorithms using low locality encoding behave like a random search.

(iii) Building Block Scaling

Encoding is uniformly scaled if all the gene values have the same contribution to the objective function, so that genetic algorithms using uniformly scaled encoding can solve all gene values implicitly in parallel. However, if some gene values have a higher contribution to the objective function than the others, this encoding is classified as non-uniformly scaled. As a result, domino convergence occurs and the gene values are solved sequentially according to their salience. The most salient gene values are solved first whereas the lowest salient gene values are solved last.

Common encoding methods for genetic algorithms often encode the phenotypes by using a sequence of gene values. The building blocks (BBs) are solved in parallel if the BBs are uniformly scaled. However, for non-uniformly scaled BBs, the BBs are solved sequentially and domino convergence occurs. Therefore, the time to convergence increases and the genetic search is affected by genetic drift. Because the

gene values are solved strictly in serial, exponentially scaled representations change the dynamics of genetic search. As a result, the solution quality is reduced by genetic drift, and the convergence time is increased by domino convergence.

The pros and cons of the five different encoding methods are listed in Table 5.5. The first one is Prüfer number encoding. Cayley (1889) identified the number of distinct spanning trees on a complete graph with n nodes as n^{n-2}. This theorem was then proven by Prüfer (1918) by the introduction of a one-to-one correspondence between spanning trees and a string of length $n - 2$ over an alphabet of n symbols. This string is denoted as Prüfer number, and the genotype-phenotype mapping is the Prüfer number encoding. It is possible to derive a unique tree with n nodes from the Prüfer number of length $n - 2$ and vice versa. Prüfer number has a number of remarkable benefits including:

(i) every tree can be represented by a Prüfer number;

(ii) every Prüfer number represents exactly one tree; and

(iii) all trees are represented equally, i.e. unbiased.

Table 5.5
Pros and cons of five common encoding methods

	Redundancy	Distance Distortion/ Locality	Building Block Scaling	Bias
Prüfer Number	Not redundant	❋ High locality around stars. ❋ Low locality elsewhere	Uniformly scaled	-
	Pros: (i) Prüfer numbers encode one-to-one mapping. (ii) Prüfer numbers represent all trees equally (unbiased). (iii) Genetic and evolutionary algorithms (GEAs) using Prüfer numbers have no problems with redundancy. (iv) The locality of Prüfer numbers representing stars is perfect. Cons: (i) Has low locality, i.e. small changes in the Prüfer number string can lead to large changes in the represented network. Thus mutation works not as a local search, but more as a random search over the solution space.			
Characteristic Vector (CV)	Uniformly redundant	High locality	Uniformly scaled	No bias
	Pros: (i) Able to represent all possible trees. Cons: (i) Can also represent non-trees: cycles in the represented graph or the graph is not connected. (ii) Can repair infeasible solutions (trees) but results in stealth mutation which increases the run duration.			
Network Random Keys (NetKeys)	Uniformly redundant	High locality	Uniformly scaled	No bias
	Pros: (i) NetKeys representation combines the advantageous elements from the LNB and CV representations. It can be interpreted that NetKeys is an improved version of the CV encoding. (ii) Standard crossover and mutation operators work properly. (iii) The encoding allows a distinction between important and unimportant links. (iv) There is no over- or under-specification of a tree possible. Cons: decoding using NetKeys is computationally expensive because it requires edge-sorting during the optimisation process.			
Direct Tree Representation (NetDir)	Not redundant	High locality	Uniformly scaled	-
	Pros: (i) The genotypic representation of the problem is just the phenotype. Therefore, the NetDir representation does not change problem difficulty and directly encodes the problem. Cons: (i) Genetic operators cannot be used any more. Proper problem-specific operators are necessary which are difficult to design. (ii) The overall difficulty of designing efficient GEAs remains the same or even increases.			
Link and Node Biased (LNB)	Pros: (i) Encoding represents the structure of a tree using a weighted vector and allows GEAs to distinguish between the importance of the nodes and links in the network. (ii) The LNB encoding becomes uniformly redundant with increasing link-specific bias. Cons: (i) Setting the link-specific bias (P_1) to zero, the representation becomes simplified node-biased encoding and it bias towards either star or minimum spanning tree (MST).			
P_1=weight of the link-specific bias, P_2=weight of the node-specific bias				
P_1=0, P_2=1	Non-uniformly redundant	High locality	Uniformly scaled	Bias towards star and MST
P_1=1, P_2=0	Non-uniformly redundant	High locality	Uniformly scaled	Bias towards MST
P_1=1, P_2=1	Non-uniformly redundant	High locality	Uniformly scaled	Bias towards star and MST
P_1=20, P_2=0	Uniformly redundant	High locality	Uniformly scaled	No bias

Because of these features, Prüfer number has been used by various researchers for the study of optimisation (Gen, *et al.*, 1999; Premkumar & Chu, 1999; Gottlieb & Eckert, 2000). The main limitation of this encoding method is its low locality for network, i.e. small changes in the Prüfer number string can lead to large changes in the represented network. Therefore, a small change made by mutation cannot work as a local search, but more as a random search over the entire solution space.

The characteristic vector (CV) is another the common approach for encoding the structure of a network (Sinclair, 1995; Tang *et al.*, 1997). A CV is a binary vector which indicates if a possible link is used or not in a network. It can represent all possible trees. Moreover, all the gene values of a CV have the same contribution to the construction of the phenotype. Therefore, there is no problem due to non-uniformly scaled gene values, domino convergence or genetic drift. Both encoding and decoding are simple. However, infeasible trees such as cycles in the represented graph or a disconnected sub-graph may also be represented. A mechanism is necessary to repair an infeasible tree back to a feasible tree. Stealth mutation will result from the repair process, which works like additional mutation and increases the run duration.

Direct tree encoding (NetDir) uses a direct representation in which no explicit genotype-phenotype mapping exists and direct representation is defined by the optimisation itself. The genotypic representation of the problem is just the phenotype. In direct representation, the trees are encoded as graph structures and not as a list of gene values. It can avoid the need for designing an efficient encoding method. Therefore, the direct representation does not change the difficulty of the problem and encodes the problem directly. However, since no standard crossover and mutation

operators can be used in direct representation, problem-specific operators have to be designed, which can be very difficult. Therefore the overall difficulty, due to the need to design an efficient algorithm, might be increased (Knowles & Corne, 2001; Rothlauf, 2002; Raidl & Julstrom, 2003).

The Link and Node Biased (LNB) encoding represents the structure of a tree using a weighted vector and allows a genetic algorithm to distinguish between the importance of the nodes and links in the network (Gaube & Rothlauf, 2001). The basic idea of the LNB encoding is to encode the importance of a node. The more important the node is, the more traffic that should transit over it. If a node is not important, the modified distance matrix should increase the length of all links that are connected to this node. However, not all the possible solution candidates can be encoded by the node-biased encoding. This can be overcome by introducing an additional link-bias in which a link-specific bias is used. Using this representation, the encoding can represent all possible trees. The performance of optimisation using the LNB encoding is good if the optimal solution is similar to stars. However, for other problems, the performance will be reduced. The major limitation of the LNB encoding is that the values of node- and link-specific bias are problem dependent. It is difficult to determine the appropriate values, especially for those who have no idea about the structure of the optimal solution.

Network Random Keys (NetKeys) encoding combines the advantageous elements from the CV and LNB representations (Rothlauf, *et al.*, 2002). It can be interpreted that NetKeys is an improved version of the CV encoding. NetKeys encoding belongs to the class of weighted encodings. The encoding allows a distinction between the importance of the links that should be used for constructing a tree. This is possible

because NetKeys encodes the importance of the links as a weighted vector while CV only encodes whether a link is established or not (Rothlauf, 2002). It is uniformly redundant and has high locality and allows the efficient use of standard crossover operators. However, the decoding using NetKeys is computationally expensive because it requires edge-sorting during the optimisation process. (Raidl & Julstrom, 2003).

Integer string representation is another type of encoding method which is similar to the CV encoding (Thiel & Voss, 1994; Savic & Walters, 1995). In integer string encoding, every possible link is assigned an integer and the presence of that link is signaled by the presence of that integer in the ordered string. The scheme for the integer assignment is arbitrary. With this integer string representation, every possible link is assigned with an integer. The number of integers should be equal to $n - 1$, where n is the number of nodes in the piping network. This kind of encoding is uniformly redundant, i.e. each phenotype is represented by the same number of genotypes so that it will not bias towards any specific structure of solution. It is simple and can be easily applied with standard crossover and mutation operators.

The basic requirements of an appropriate encoding method used in genetic algorithm for the present study are simplicity, ability to represent all the possible piping configurations and non-bias to any type of piping configuration. Direct tree encoding is simple as the genotypic representation is exactly the phenotype. However, difficult and complicated problem-specific operators have to be designed. For Link and Node Biased encoding method, the major limitation is that it biases towards a certain type of piping configuration, which is influenced by weighting factors of link-specific bias and

node-specific bias. There is no bias in using Network Radom Keys encoding method. However, this method is computationally expensive as it requires edge-sorting during the optimisation process.

The remaining encoding methods including Prüfer Number, Characteristic Vector and Integer String are simple and capable to encode all the possible piping configurations without bias. The locality of Prüfer Number is low while the Characteristic Vector and Integer String cannot avoid representing infeasible piping configurations which require repairing. In order to compare their performance and select an appropriate one for the present study, a set of experiments over 20 runs for a DCS piping network of size 9 was carried out for two encoding methods, Prüfer Number and Integer String (since the features of Characteristic Vector are very close to that of Integer String, only the latter is involved in this comparative test).

Optimisation of the DCS piping configuration was carried out using genetic algorithm at different crossover and mutation rates. The values of the rates have influence over the ability of the search to find the optimum. A smaller population size, lower probability of crossover and lower mutation rate would result in faster convergence, but would reduce the degree of randomness of the search. This may result in failure to find the optimum solution. On the other hand, increasing the population size would slow convergence and a mutation rate over 10% would result in a purely random search (Wright, 1996). There is a need to choose the parameter values to balance the variety of individuals evaluated against the rate of convergence. With reference to the values used by other researchers for similar combinatorial optimisation problems; and types of crossover and mutation, suitable ranges of the values for crossover and mutation rates

are 0.3 to 0.8; and 0.01 to 0.2, respectively; and the population size commonly used is 20 (Morley, 2001; Farmani, *et al.*, 2005; Obara, 2006; Bahadoorsingh, *et al.* 2007; Kessels, *et al.*, 2007).

In this comparison, a population size of 20; crossover rates of 0.1, 0.2, 0.4 and mutation rates of 0.01, 0.1, 0.2 were used in these experimental runs. The results are listed in Table 5.6. It was found that the Integer String encoding method can find the minimum function value (1,338 x 10^6) under all crossover and mutation settings. However, Prüfer Number encoding method can only search this minimum value at crossover rate of 0.1 and mutation rate of 0.01. In terms of average function value, the results obtained by using Integer String were also superior to those obtained using Prüfer Numbers. The best value was found under a crossover rate of 0.4 and mutation rate of 0.1.

Table 5.6
Results over 20 runs for problem of size 9 for encoding methods: Prüfer No. and Integer String

	Prüfer No.	Integer String	Prüfer No.	Integer String	Prüfer No.	Integer String
	Crossover Rate: 0.1		Crossover Rate: 0.2		Crossover Rate: 0.4	
Mutation Rate: 0.01	1,338 (1,535)	1,338 (1,521)	1,366 (1,682)	1,338 (1,455)	1,366 (1,686)	1,338 (1,478)
Mutation Rate: 0.1	1,370 (1,563)	1,338 (1,412)	1,410 (1,617)	1,338 (1,436)	1,451 (1,708)	1,338 (1,382)
Mutation Rate: 0.2	1,370 (1,387)	1,338 (1,472)	1,451 (1,704)	1,338 (1,425)	1,529 (1,747)	1,338 (1,425)

* In each cell, the first value is the minimum function value found.
In each cell, the second value (inside the parentheses) is the average function value found.
** All function values are in (x 10^6)

As a result, the Integer String encoding method is selected to represent the DCS piping configuration in this study. An example of a piping configuration of problem size 9 with Integer String representation is illustrated in Figure 5.10 and Table 5.7. For a piping network with n nodes, the number of integers generated for each population (candidate) should be equal to $n - 1$. In Figure 5.10, node 1 is a DCS central plant which supplies chilled water to the consumer buildings (nodes 2 – 9) through a piping network. This configuration is encoded by an integer-string {1, 5, 8, 9, 23, 27, 34, 35} as listed in shaded cells of Table 5.7. The corresponding values of CV labels of this example piping network are also listed in Table 5.7. Each edge in the piping network represents two parallel pipes including the supply and return pipes.

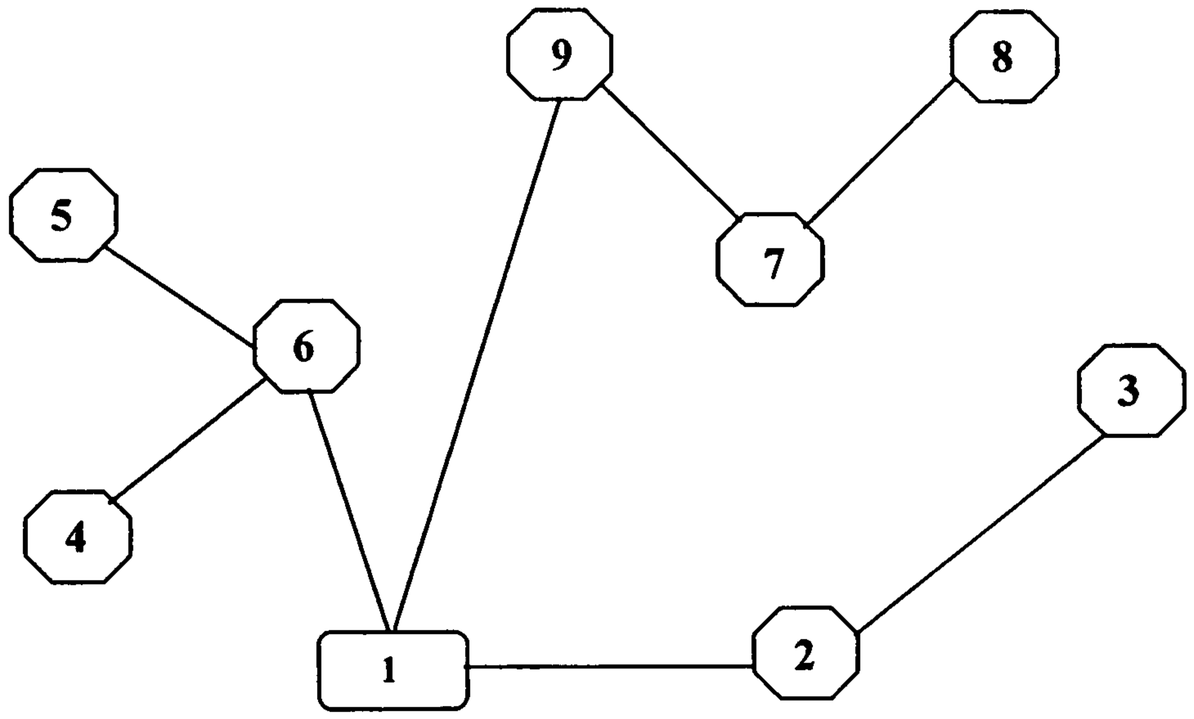


Figure 5.10 An example piping network with integer-string {1, 5, 8, 9, 23, 27, 34, 35}

Table 5.7
Integer string and CV encodings for an example piping network

CV label	Integer label	Link
1	1	1-2
0	2	1-3
0	3	1-4
0	4	1-5
1	5	1-6
0	6	1-7
0	7	1-8
1	8	1-9
1	9	2-3
0	10	2-4
0	11	2-5
0	12	2-6

CV label	Integer label	Link
0	13	2-7
0	14	2-8
0	15	2-9
0	16	3-4
0	17	3-5
0	18	3-6
0	19	3-7
0	20	3-8
0	21	3-9
0	22	4-5
1	23	4-6
0	24	4-7

CV label	Integer label	Link
0	25	4-8
0	26	4-9
1	27	5-6
0	28	5-7
0	29	5-8
0	30	5-9
0	31	6-7
0	32	6-8
0	33	6-9
1	34	7-8
1	35	7-9
0	36	8-9

5.3 Local Search and Looped Local Search

The basic format of a combinatorial optimisation problem can be expressed as:

$$f(s) = \min\{f(x) | \in S\} \tag{5.2}$$

The set S of feasible solutions is finite and the objective is to find an element s in S which has a minimum objective function. Since the cardinality of S is extremely large, it is impossible to search the optimal solution exhaustively. In order to improve the solution quality and computational efficiency, local search technique was incorporated into the GA. The basic idea of local search is to start with some initial solution and move from neighbour to neighbour as long as possible while the objective value is decreasing. The choice of neighbourhoods is a compromise between the solution quality and complexity of the algorithm. It is because small neighbourhoods are easy

for searching but offer little room for improvement while large neighbourhoods may be very tedious to explore.

One of the important issues in implementing a local search in combinatorial optimisation is how to define and select the neighbourhood structure of a given problem. The following paragraph shares some experience by various researchers.

Dengiz *et al.* (1997) developed and applied a new local search operator to optimise the design of communication network topology. The communication network was modelled by a probabilistic graph in which nodes and links corresponded to the computer sites and communication connections. The networks were assumed to have bidirectional links and therefore were modelled by graphs with non-directed links. The objective function was the sum of the total cost for all the links in the network. Initial population was generated by the depth-first search algorithm (Hopcroft & Ullman, 1973) which grew a spanning tree from randomly chosen nodes. An elitist ranking selection with stochastic remainder sampling without replacement was used to select two candidate networks for crossover and mutation. Crossover was a form of uniform crossover with repairing to ensure that each offspring was at least a spanning tree with two-connectivity. Mutation took the form of a randomised greedy local search operator. The links of the communication network were ranked in descending order. The maximum cost link was selected and dropped from the network. If the network still had two-connectivity, the local search stopped. Otherwise, the local search was repeated for the remaining ranked links until one was dropped or the list exhausted. The results from this algorithm outperformed the optimum results found by branch & bound technique and the GA without local search.

A biased local search operator was developed by Raidl *et al.* (2006). In the mutation process, an edge was selected for removal according to one of the following three selection strategies:

(i) the edge was chosen uniformly at random;

(ii) the edge chosen was the edge of largest weight;

(iii) the edge was chosen according to the probability derived from the rank assigned (in ascending order of their weights).

Then the mutation operator selected and inserted a new edge into a feasible solution according to one of the following strategies:

(i) the edge was chosen randomly with a probability which was inversely proportional to its weight;

(ii) the edge was chosen based on a normal distribution.

The authors derived weight-biased mutation in evolutionary algorithm and investigated empirically the distributions of edges in optimum solutions of the minimum spanning tree problem, degree constrained minimum spanning tree problem and travelling salesman problem, in terms of the edges' weighted-based ranks. The performance of various edge-selection strategies were studied and compared.

For a topology design and capacity assignment problem, Sayoud *et al.* (2001) hybridised a GA with a hill-climbing local search to combine the strengths of both by providing global and local exploitation aspects to the topology problem. During the optimisation process, an elite nondominated set of solutions was selected. An

intermediate population which was composed of the elite and the improved solutions by genetic operators was constructed and then a Nelder and Mead simplex downhill method was applied to some solutions. Improved results were obtained by combining the advantage of global exploitation from the GA and local exploitation of local search.

Yamada and Nakano (1996) integrated multi-step crossover and a local search method and applied this combination to a job-shop scheduling problem. In the approach, a solution initially set to be one of the parents was stochastically replaced by a relatively good solution from the neighbourhood, where the replacement was biased towards the other parent. The procedures were: firstly, all the members in the neighourhood were indexed in ascending order according to the distance from the other parent. Then a member was selected from the neighbourhood randomly, but a smaller index was preferred. It was then probabilistically accepted according to its evaluation value. This technique was applied to a job-shop scheduling problem which employed a critical path-based neighbourhood, i.e. the longest weighted path from source to sink in a graph.

In a network reconfiguration problem, Brown (2003) developed a novel algorithm, annealed local search, to adjust switch positions until an optimal solution was identified. The annealed local search avoided the problems of simulated annealing and genetic algorithms by making small changes in the radial structure of a distribution system by closing a normally open switch and opening a nearby upstream switch. The process, referred to as a tie switch shift, allowed incremental changes to the tree structure as opposed to complete reformulation of the structure for each switch position change.

Ishibuchi *et al.* (1997) found that, in performing a local search with genetic algorithm,

all the neighbourhood solutions of the current solution should be examined before the local search procedure is terminated. This resulted in the number of generations in the optimisation process becoming small since almost all the computation time was spent in the local search procedure. In order to improve this area, they developed a genetic local search (GLS) algorithm which was a hybrid algorithm of a local search and a genetic algorithm. It was applied to a flow shop scheduling problem in which, if a certain number of neighbourhood solutions of the current solution had been examined and there was no better solution among the examined neighbourhood solutions found, the local search procedure for the current solution would be ended. The result showed that this GLS technique outperformed the simple GA and the multi-start local search (multi-start hill-climbing).

In a quadratic assignment problem, Merz and Freisleben (1997) applied a 2-opt heuristic, also known as the pair-wise interchange heuristic, as the local search method. Two elements were selected and swapped in the permutation. This approach had been successfully used to solve large instances of the travelling salesman problem. Merz and Freisleben indicated that this genetic local search was also applicable to the quadratic assignment problem since the algorithm was able to arrive at high quality solutions in a short time, especially for instances whose solution space had structures typically found in real life problems.

The above-mentioned investigations on local search have a variety of features and applications. The approach developed by Dengiz *et al.* (1997) only selected and dropped the link of a communication network with maximum cost link. There was no method developed in the local search algorithm for searching any new link in the

neighbour of the current solution. In the study by Raidl *et al.* (2006), the new link was selected either based on a normal distribution or on a probability inversely proportional to its weight. The original two nodes connecting the deleted link might not be involved in the new link. This could lead to a large change in the minimum spanning tree configuration during the local search process. Therefore a better solution in the vicinity of the current near-optimal solution might be missed. The local search procedures by Sayoud *et al.* (2001) chose a number of best solutions from the population and then selected randomly a neighbourhood. If the function value of this neighbourhood improved, it replaced the current near-optimal solution. However, the structure of the neighbourhood was not well defined and stated in Sayoud *et al.*'s paper. In Yamada and Nakano's approach (1996), all members in the neighbourhood along a critical path of a schedule were indexed in ascending order and the member was selected randomly from this neighbourhood. Again, a better solution in the vicinity of the current near-optimal solution might be missed. In Brown's novel algorithm (2003), a tie switch was randomly selected and local search was carried out in either a forward or backward direction. Since there were no constructive heuristics for selecting the tie switch, this local search method might not be effective. The genetic local search algorithm proposed by Ishibuchi *et al.* (1997) was only a method for reducing the computation time spent by local search. No new idea was developed for carrying out a local search for combinatorial problems. Merz and Freisleben's 2-opt pair-wise interchange heuristic (1997) was actually one form of crossover operator applied into two selected individuals.

The previous local search algorithms developed by various researchers are not effective nor suitable for being applied directly to the current piping configuration optimisation

problem. There was therefore a need to develop a new and appropriate type of local search technique for the optimisation of the piping configuration in DCS.

A local search can be applied to each individual candidate in every generation, but this is time consuming and may not be effective. Therefore, in the present study, a local search is applied if a local minimum is found, i.e., there is no further improvement in the function value after a certain number of generations. The elite is then taken as the candidate for local search. Two types of local search were developed, namely Local Search and Looped Local Search, for the present optimisation problem (Chan *et al.*, 2007a). The steps of Local Search/Looped Local Search for problem of size m are detailed as follows:

Local Search

(i) If the function value converges and there is no further improvement by GA operators (crossover & mutation) after x generations (arbitrarily assigned), subroutine "Local Search/Looped Local Search" will be called.

(ii) The elite in the current generation will be taken as the candidate for Local Search/Looped Local Search.

(iii) The longest link ($link_{longest}$) of the elite and the two nodes ($node_{NB}$ and $node_{NE}$) joining this link are identified.

(iv) Take $node_{NB}$ as a searching point and connect to the other $node_j$, ($j \in \{1, 2, ..., m\}$, $j \neq$ NB or NE), to form a new $link_{NB-j}$.

(v) The new link: $link_{NB-j}$ replaces the original longest link, $link_{longest}$, to form a new piping network.

(vi) Calculate the function value of this new network.

If function $value_{NB-j} \geq$ function $value_{elite}$, do nothing.

If function $value_{NB-j} <$ function $value_{elite}$, assign a new best function value: function $value_{elite} \leftarrow$ function $value_{NB-j}$ and elite $\leftarrow$ new solution.

(vii) Repeat steps (iv) – (vi) using other nodes of j ($j \in \{1, 2, ..., m\}$, $j \neq$ NB or NE) until $j = m$.

(viii) Repeat steps (iv) – (vii) using $node_{NE}$ as the searching point.

Looped Local Search

(ix) If there is no further improvement by GA operators (crossover & mutation) and Local Search after x generations, the second longest link $(link_{k,\ longest})_{(k=2^{nd})}$ of the elite and the two nodes ($node_{NB,k}$ and $node_{NE,k}$) joining the link are identified.

(x) Take $node_{NB,\ k}$ as a searching point and connect to the other $node_j$, ($j \in \{1, 2, ..., m\}$, $j \neq NB_{,k}$ or $NE_{,k}$), to form a new $link_{NB,k-j}$.

(xi) The new link: $link_{NB,k-j}$ replaces the original link, $link_{k,\ longest}$, to form a new piping network.

(xii) Calculate the function value of this new network.

If function $value_{NB,k-j} \geq$ function $value_{elite}$, do nothing.

If function $value_{NB,k-j} <$ function $value_{elite}$, assign a new best function value: function $value_{elite} \leftarrow$ function $value_{NB,k-j}$ and elite $\leftarrow$ new solution.

(xiii) Repeat steps (x) – (xii) using other nodes of j ($j \in \{1, 2, ..., m\}$, $j \neq NB_{,k}$ or $NE_{,k}$) until $j = m$.

(xiv) Repeat steps (x) – (xiii) using $node_{NE,k}$ as the searching point.

(xv) Repeat steps (ix) – (xiv) for k = (3^{rd}, 4^{th}, ……., $m - 1^{th}$), i.e. the third longest link, fourth longest link, ……, $m - 1^{th}$ longest link.

An illustrative example of the Local Search/Looped Local Search is shown in Figures 5.11 (a) – (g). As seen from Figure 5.11 (a), the link 1 – 9 is the longest link ($link_{longest}$). Node 9 ($node_{NB}$) is identified as the searching point. The original longest link 1 – 9 is deleted and a new link 9 - 2 ($link_{NB-j,\ j=2}$) is formed to construct a new piping network (see Figures 5.11 (b) & (c)). If the function value of this new network is lower than that of the elite, this new network will replace the elite. The Local Search continues for other $node_j$ ($j = 3, ..., 8, j \neq 9$), that is the new links 9 – 3, 9 – 4, ……, 9 – 8 are formed, respectively, as shown in Figures 5.11 (d), (e) & (f). For each network with a newly formed link, the function value is calculated and compared with that of the current elite (infeasible networks are detected and repaired as discussed in Section 5.1). If the function value of this new network is smaller than that of the elite, this new network will replace the elite. After that, another search point, node 1 will be used to perform the similar local search processes as described above. For Looped Local Search, the second longest link $(link_{k,\ longest})_{(k=2^{nd})}$: 2 – 3 is located (see Figure 5.11 (g)). $Node_{NB,k} = 2$ and $node_{NE,k} = 3$ are identified and a similar local search performed until the shortest link $(link_{k,\ longest})_{(k=8^{th})}$ has undertaken the last local search process.

Through these Local Search and Looped Local Search processes, better solutions can be found for in the vicinity of the current near-optimal solution. This can improve the effectiveness and efficiency of the optimisation process conducted by GA for the piping configuration in a DCS.

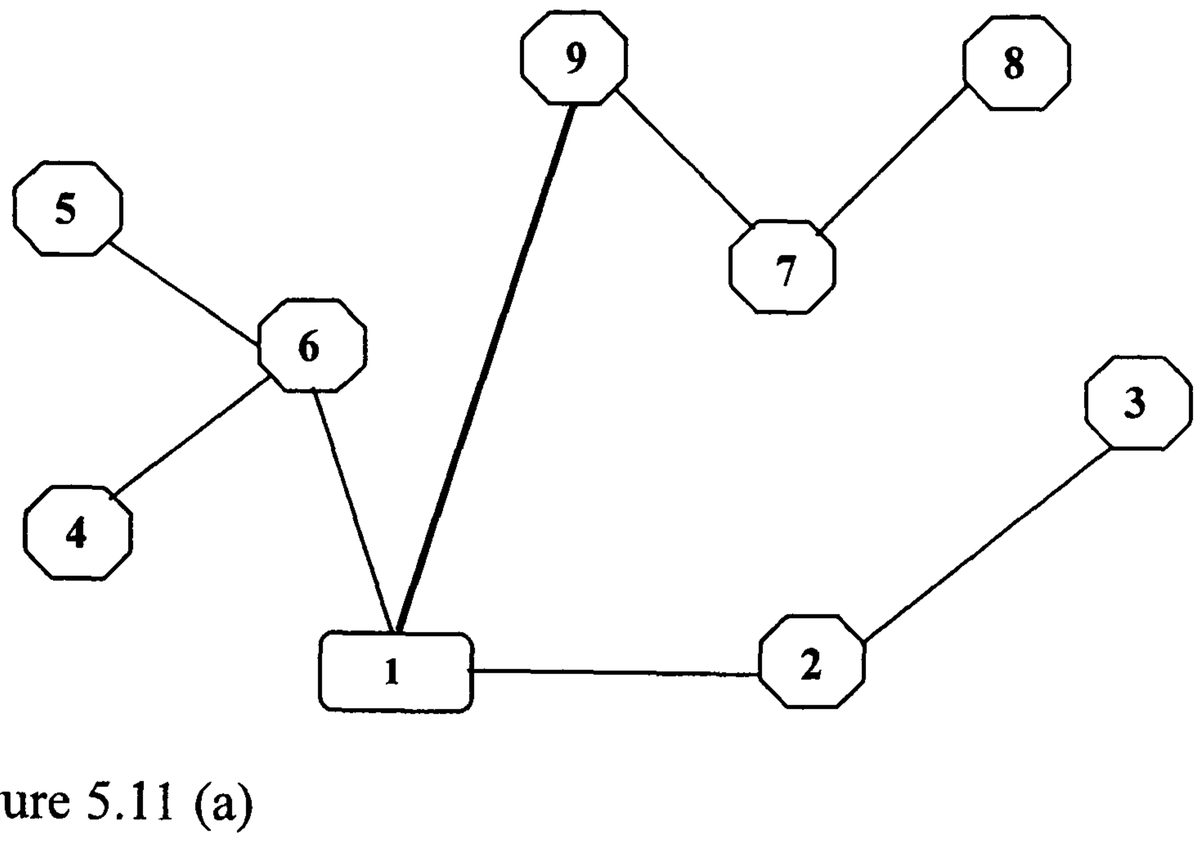


Figure 5.11 (a)

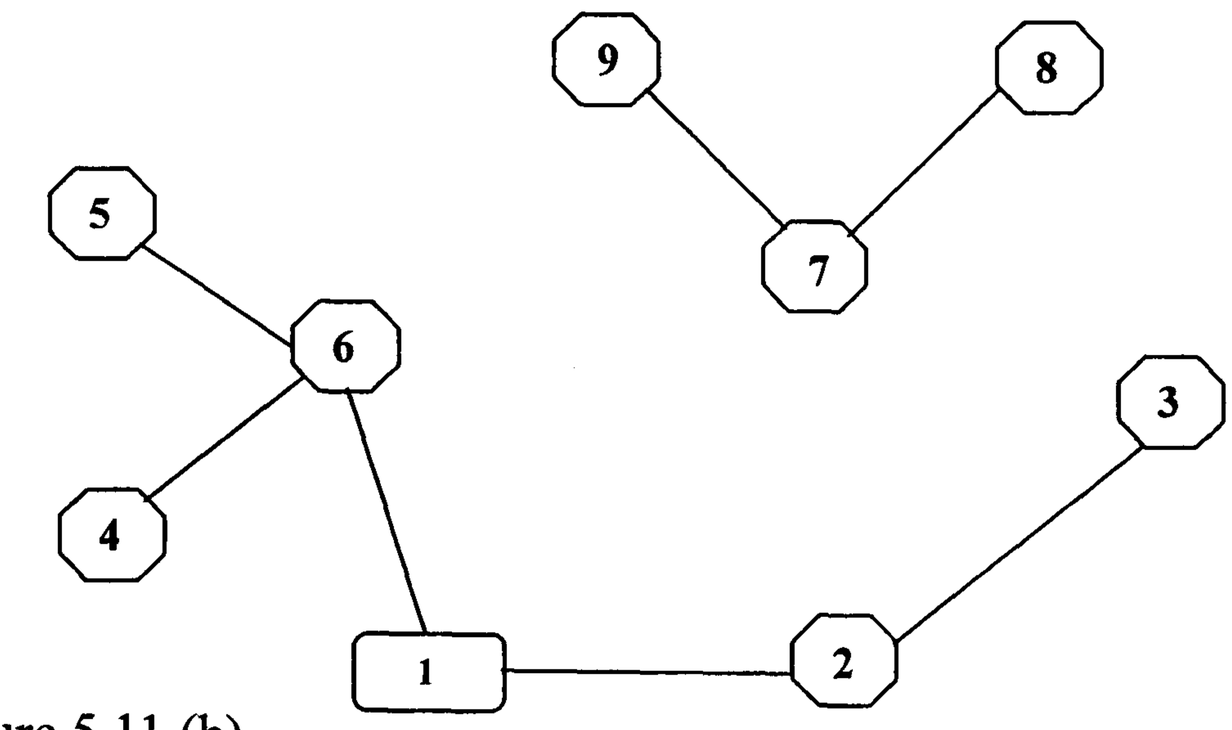


Figure 5.11 (b)

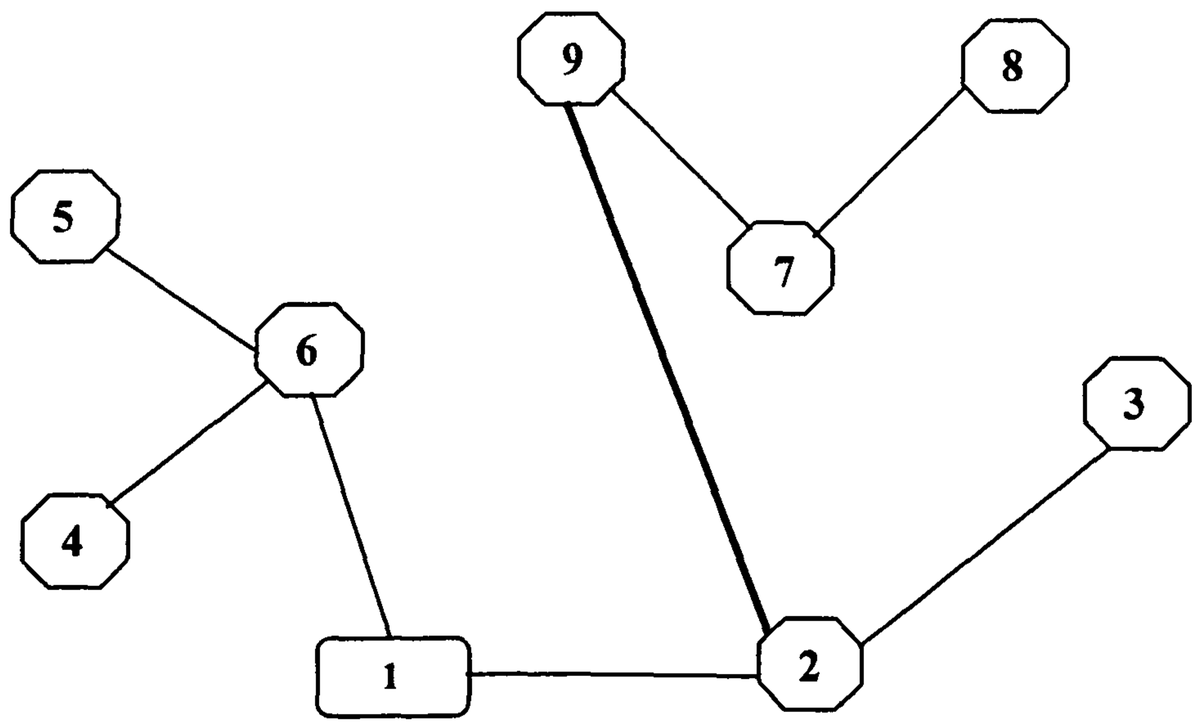


Figure 5.11(c)

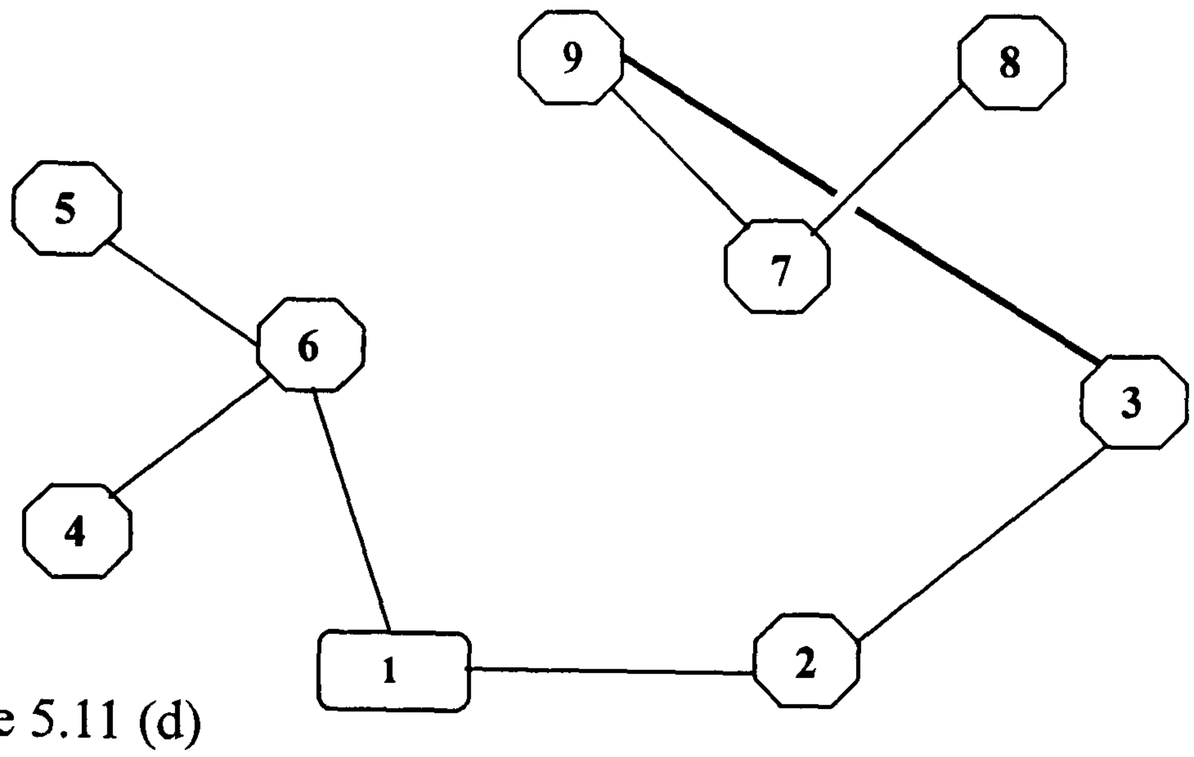


Figure 5.11 (d)

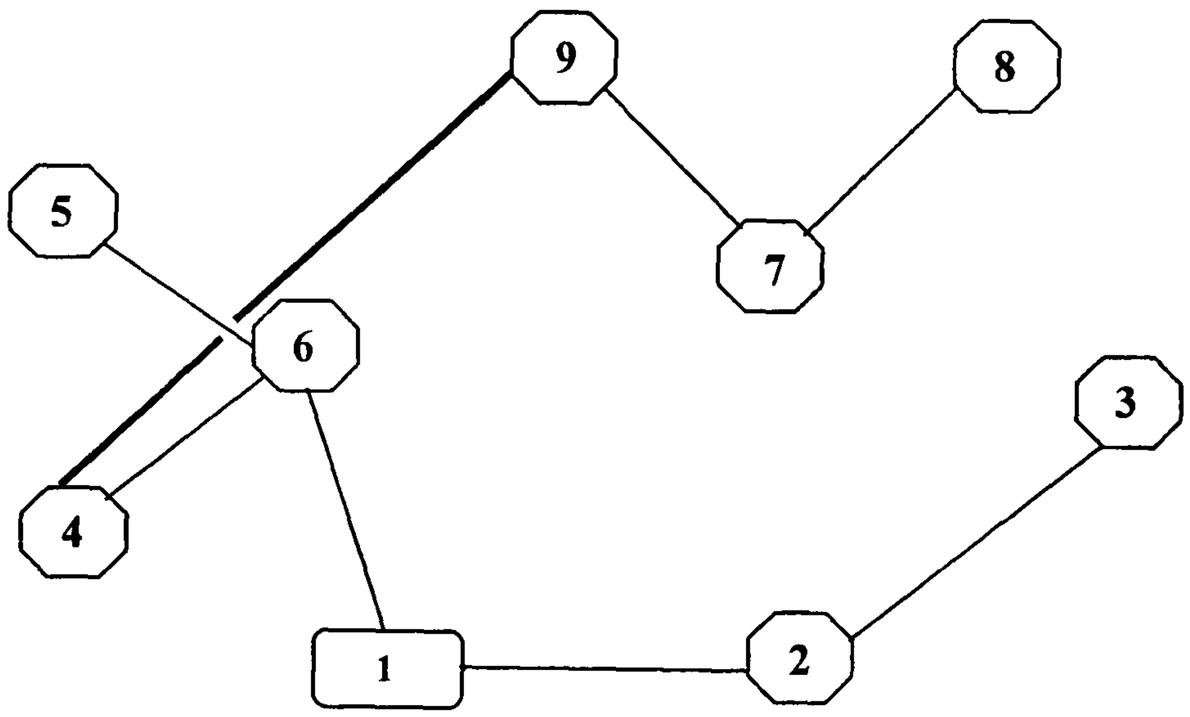


Figure 5.11 (e)

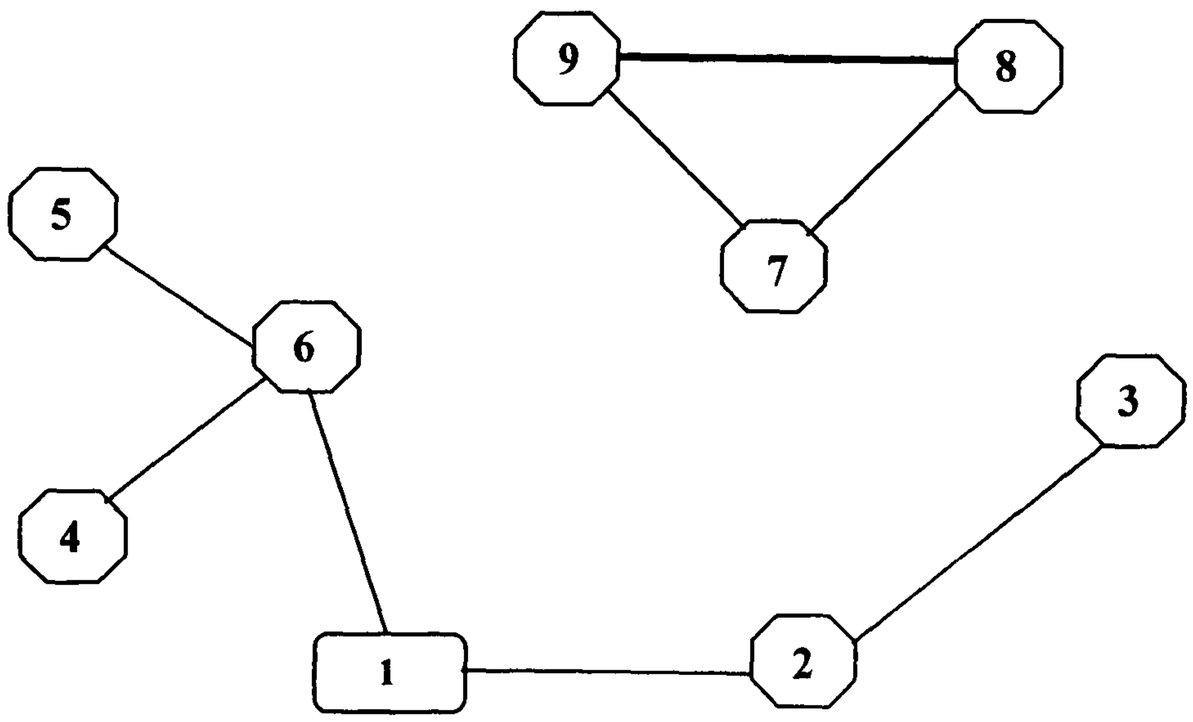


Figure 5.11 (f)

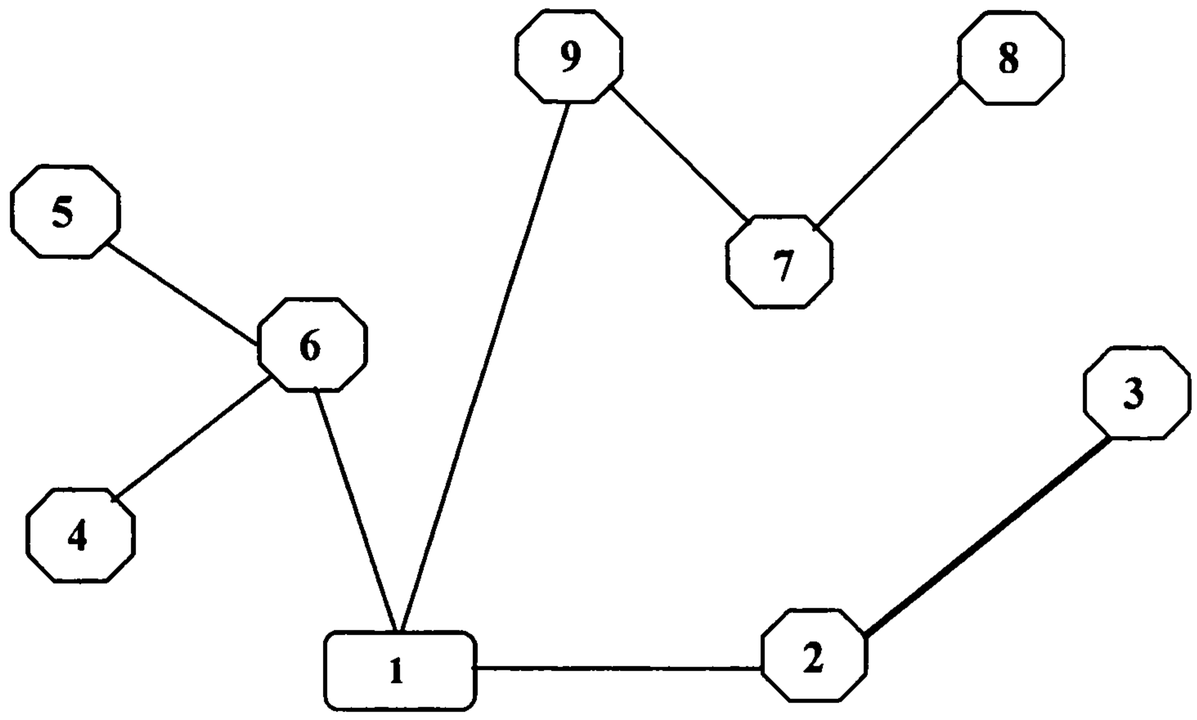


Figure 5.11 (g)

Figure 5.11 Illustrative example of Local Search/Looped Local Search

In this chapter, through a detailed review, an appropriate type of encoding method was selected for the present study. With the designed methodology and local search techniques developed, the impact of various factors on the performance of GA in optimisation was investigated through a series of computational experiments, which are detailed in next chapter. Full understanding of the impact of those factors is crucial for the optimal design of piping distributing piping network in DCS.

CHAPTER 6
PARAMETRIC Experiments AND ANALYSIS

For effective optimisation of the distribution piping network in a district cooling system, in terms of computational time and solution quality, it is essential to investigate the best way to carry out the optimisation process. Since genetic algorithms consist of various components and parameters as well as how the local search technique is incorporated and implemented, it is crucial to understand the impact of these factors on the performance of optimisation carried out by GA. Although there are previous studies on the impact of some factors including population size, encoding method, reproduction operators, etc. on the performance of GA, especially in continuous optimisation problems, little research has examined the effect on the optimal design of distribution piping network in a district cooling system. In this chapter, a study of the effects of various types of major factors and the application of local search in GA for the optimisation of distribution piping networks is presented and discussed.

6.1 Local Search

GA performs a global sampling of the search domain. This type of heuristic algorithm for solving optimisation problems can find near-optimal solutions within a reasonable amount of computational time. Better solutions in the vicinity of the current near-optimal solution may be missed if the search process just relies on probabilistic search heuristics. In order to avoid missing and to search those possible better solutions, a local search algorithm was developed and incorporated into GA to find better solutions, generally starting with a feasible solution found by the crossover/mutation operators. At each iteration, an improving solution could be found

by searching the neighborhood of the current solution.

Local search can be applied to each individual candidate in every generation, but this is time consuming and may not be effective. Therefore, in the present study, local search is applied if a local optimum is found, i.e., there is no further improvement for the objective function after a certain number of generations. The elite is taken as the candidate for local search. As reviewed and investigated in Section 5.2, the crossover rate and mutation rate used in this study are 0.4 and 0.1, respectively. Since there is no convergence criterion specified for the piping network optimisation problem in the present study, the search was run for 200 generations which was selected by inspection (see Figure 6.1 below) to ensure that convergence was reached. A maximum number of generation, 200, is used as a termination criterion for each run.

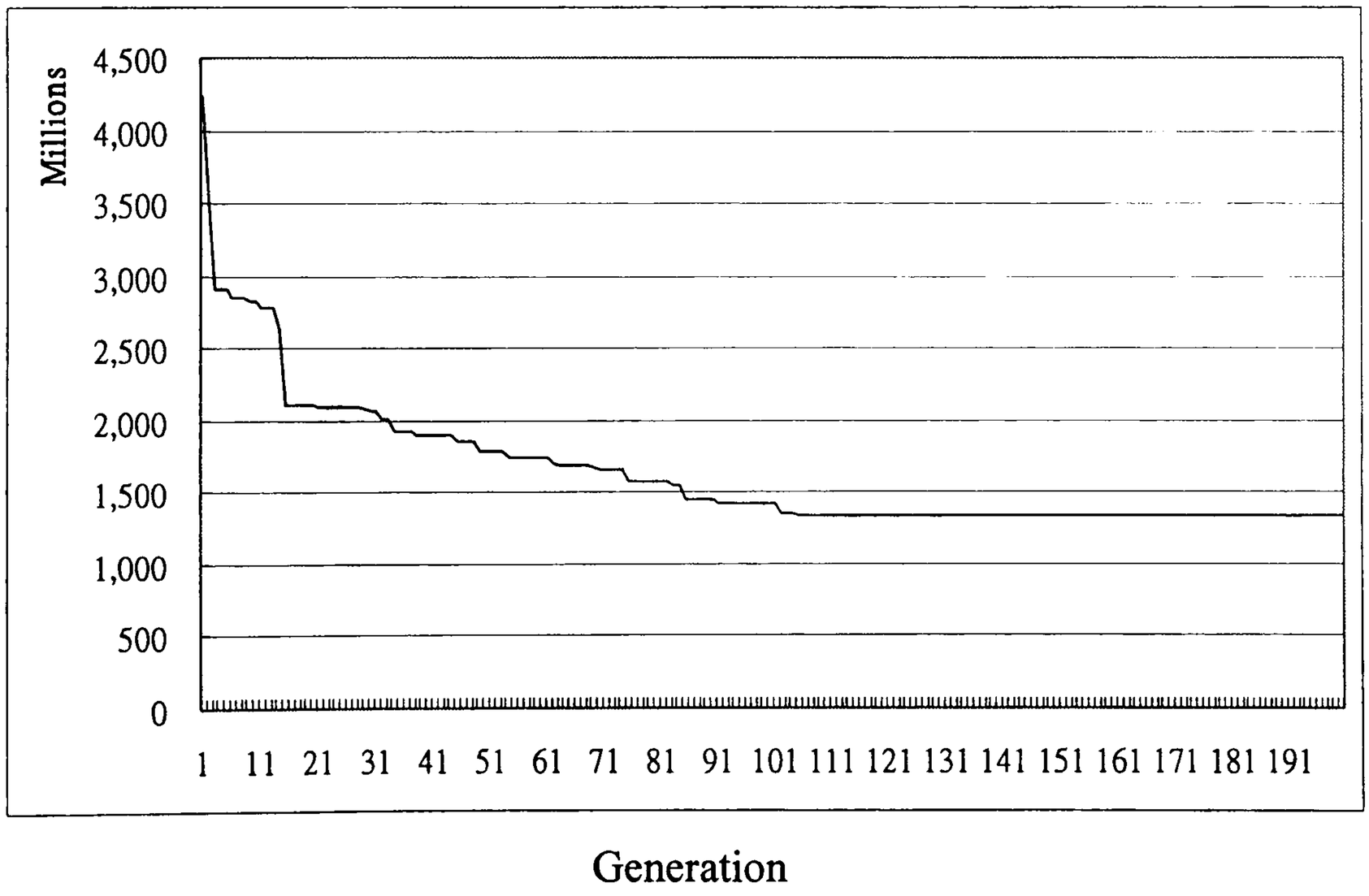


Figure 6.1 Optimal solution in each generation for problem of size 9

Table 6.1 lists the results of 20 runs for a case of a central district cooling plant with 8 building groups. In Table 6.1, the left main column shows the experimental results for the case without local search (base case) while the one on the right column is the results for the case with Local Search. Two performance indices, namely the best function value and the first hit generation number (the generation number at which the best solution first appears), are presented. Instead of using generation number as a measure of the computational effort taken in the optimisation process, the total number of objective function evaluations can be an alternative choice in which the computational effort associated with local search is counted. From the experimental runs in this study, it was found that the extra function calls of the local search only account for 2 to 3% of the total function evaluations. Moreover, for easy comparison with the published results from other researchers, generation number was used in this study. In addition to the results of the 20 runs, minimum value, average value, standard deviation (Std Dev) and success rate (Succ R) are also listed for analysis.

It can be seen that both cases can find the reasonable near-optimal solution ($1{,}338 \times 10^6$) which was confirmed by inspection, as discussed above with Figure 6.1. For both the average value and the first hit generation number, the case with Local Search outperforms the base case. The average values obtained by Local Search are $1{,}356 \times 10^6$ and 116, respectively, which are 3.0% and 14.1% better than the base case. In terms of successful searching for the optimal solution, the base case has achieved a value of 50% (success rate) while the value for the case with Local Search is 80%. A large standard deviation indicates that the data points are far from the average and a small standard deviation indicates that they are clustered closely around the average. The standard deviation of the best function value for the case with Local Search is 43.9

x 10^6 while that for the base case is 63.0 x 10^6. These two values are small when compared with the corresponding average values. For the first hit generation number, the standard deviation of the case with Local Search is 36.3 and that for the base case is 31.5. Both values are close to each other.

Table 6.1
Experimental results over 20 runs for problem of size 9 without and with Local Search

	Without Local Search		With Local Search	
Run no.	Best Function Value ($x10^6$)	First Hit Generation No.	Best Function Value ($x10^6$)	First Hit Generation No.
1	1,338	121	1,338	105
2	1,478	82	1,431	82
3	1,431	150	1,338	177
4	1,338	156	1,338	112
5	1,338	71	1,338	71
6	1,338	132	1,338	61
7	1,338	182	1,339	165
8	1,444	193	1,338	188
9	1,443	142	1,338	98
10	1,459	139	1,338	76
11	1,338	116	1,478	119
12	1,338	158	1,338	167
13	1,478	168	1,338	135
14	1,431	143	1,338	92
15	1,338	159	1,338	88
16	1,478	108	1,459	136
17	1,459	135	1,338	119
18	1,478	92	1,338	127
19	1,338	141	1,338	93
20	1,338	119	1,338	111
Min.	1,338	71	1,338	61
Average	1,398	135	1,356	116
Std Dev	63.0	31.5	43.9	36.3
Succ R	50%	-	80%	-

Although the case with Local Search gives better performance, the effect of local search is not very significant for this case of small problem size. In order to investigate and illustrate the significant effect of local search in piping network optimisation, the problem size is increased from 9 to 17 (i.e. one central DCS plant serving 16 scattered consumer buildings). Similar to the problem case of size 9, the termination criterion is determined by inspection to ensure reaching convergence (see Figure 6.2 below). The value used is 1,000 runs. The experimental results over 20 runs are listed in Table 6.2 for analysis. As expected, the case with Local Search for the problem of size 17 outperforms the base case, as shown by an averaged best function value of 3,535 x 10^6, which is 16.2% better than the base case. The superior effect of local search is more significant for the problem of size 17 with increased complexity. Again this is the result of local search to avoid missing of better solution (3,214 x 10^6). The average first hit generation number is reduced to 708 as compared to the base case (890). However, when compared with the simpler case of size 9, the success rate is only 15% in this case which is not satisfactory. Areas for improvement will be addressed in the coming sections.

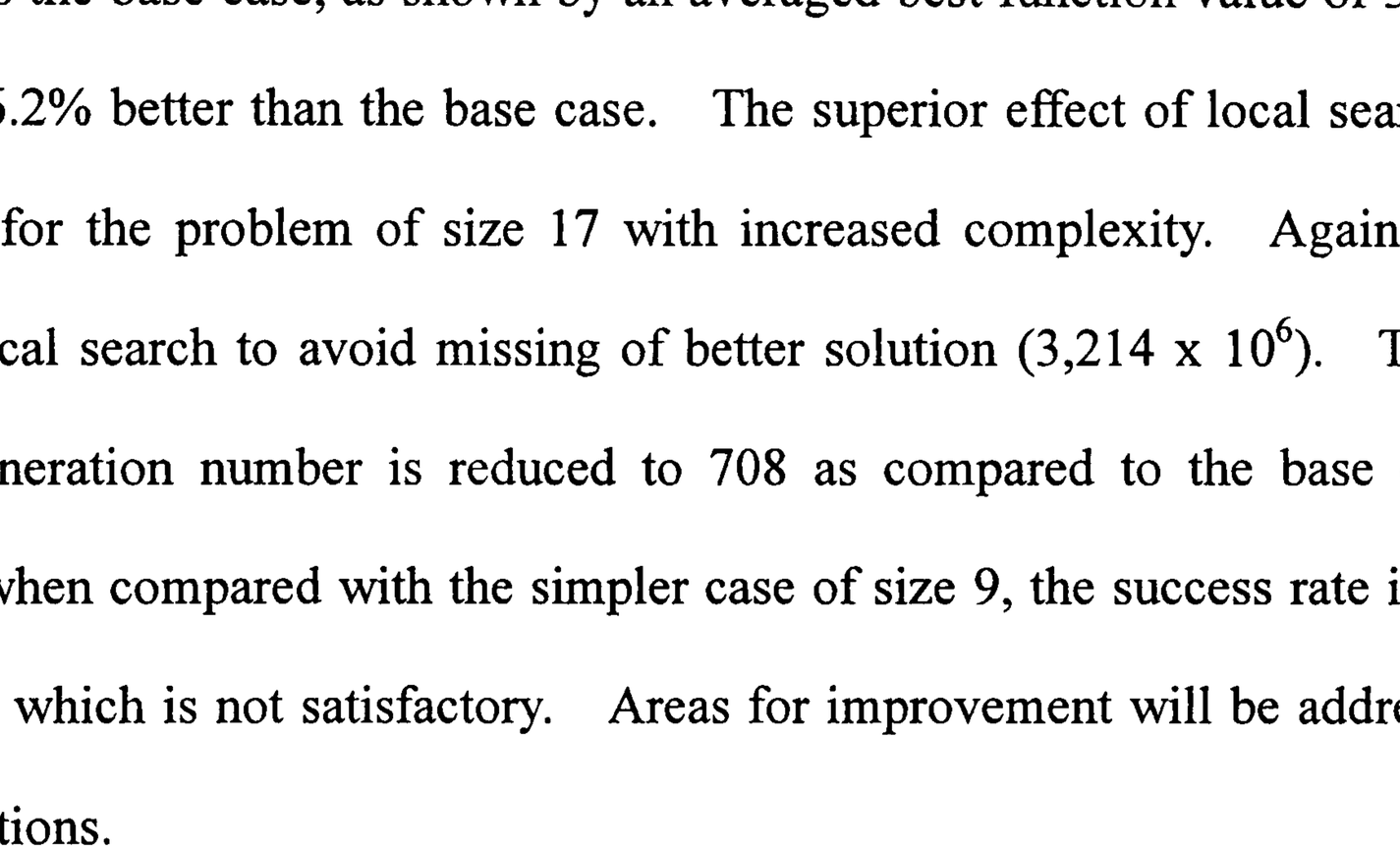


Figure 6.2 Optimal solution in each generation for problem of size 17

Table 6.2
Experimental results over 20 runs for problem of size 17 without and with Local Search

Run no.	Without Local Search		With Local Search	
	Best Function Value ($x10^6$)	First Hit Generation No.	Best Function Value ($x10^6$)	First Hit Generation No.
1	3,577	935	3,698	418
2	3,290	772	4,100	717
3	3,592	896	4,033	434
4	3,656	970	3,547	766
5	5,077	976	3,360	960
6	4,453	907	3,257	648
7	4,790	614	3,406	602
8	4,204	998	3,392	785
9	4,748	914	3,214	769
10	4,699	929	3,360	990
11	3,592	768	3,406	698
12	4,204	925	3,406	749
13	4,790	915	4,204	954
14	3,656	836	3,698	825
15	4,699	914	3,547	841
16	4,204	869	3,214	592
17	3,577	787	3,656	681
18	3,290	953	3,392	636
19	4,204	958	3,214	499
20	4,790	963	3,592	587
Min.	3,290	614	3,214	418
Average	4,219	890	3,535	708
Std Dev	580.3	93.2	292.4	162.5
Succ R	0%	-	15%	-

6.2 Local Search Versus Looped Local Search

In Section 6.1, it was found that the performance of the GA using local search is better than that without local search. In the Local Search loop, the longest pipe segment of the distribution piping network is identified and replaced by newly formed link (pipe segment). However, there is no rule that links other than the longest one, if selected,

cannot perform better in the local search. In order to investigate this point and further enhance the performance of GA, a Looped Local Search approach was developed. The details of the Looped Local Search algorithm were presented in Chapter 5 (Section 5.3). The comparison of the experimental results for this case is listed in Table 6.3.

Table 6.3
Experimental results over 20 runs for problem of size 17 with Local Search and Looped Local Search

	With Local Search		With Looped Local Search	
Run no.	Best Function Value ($x10^6$)	First Hit Generation No.	Best Function Value ($x10^6$)	First Hit Generation No.
1	3,698	418	3,305	446
2	4,100	717	3,225	871
3	4,033	434	3,599	334
4	3,547	766	2,913	395
5	3,360	960	3,395	259
6	3,257	648	2,913	897
7	3,406	602	3,080	460
8	3,392	785	2,913	866
9	3,214	769	3,178	776
10	3,360	990	3,131	310
11	3,406	698	2,913	498
12	3,406	749	3,225	725
13	4,204	954	2,913	662
14	3,698	825	2,913	595
15	3,547	841	3,178	398
16	3,214	592	3,599	784
17	3,656	681	3,599	558
18	3,392	636	2,913	698
19	3,214	499	3,225	368
20	3,592	587	3,080	411
Min.	3,214	418	2,913	259
Average	3,535	708	3,161	566
Std Dev	292.4	162.5	241.5	205.4
Succ R	0%	-	35%	-

The results are presented in a similar format as that in Section 6.1. It is encouraging to note that, with Looped Local Search, there is further improvement in the optimal solution found ($2{,}913 \times 10^6$). The success rate increased to 35%. The average best function value is $3{,}161 \times 10^6$ which is 10.6% better than the case with Local Search only. This demonstrates that links other than the longest one can contribute to the local search for finding better solution in the neighbourhood of the current near-optimal piping configuration.

6.3 Repairing Algorithm Versus Penalty Value

In the present study, the Integer String method was used to encode the piping circuit. As a feature of this encoding method, infeasible candidates such as a case with non-connected node(s); with repeated link(s) and/or with looped circuit(s) may be generated. Killing an infeasible candidate is not recommended since a good solution may be the result of breeding from feasible and infeasible candidates. For treating the infeasible candidates, two major approaches can be used, including repairing the infeasible candidate and incorporating a penalty value into the objective function in order to retain the infeasible candidate in the population.

For repairing an infeasible candidate, the following procedures were used:

(i) Identify disconnected node(s) which is (are) caused by repeated link(s) and/or loop(s).

(ii) Identify a repeated link and then erase this repeated link. Connect one of the nodes, which originally connected the repeated link, to the disconnected node. Thereby the disconnected node can be connected to the piping circuit.

(iii) If there is no repeated link in (ii), identify a loop and then erase one of the links in the loop. Connect one of the nodes, which originally connected the link of the loop, to the disconnected node. Thereby the disconnected node can be connected to the piping circuit.

(iv) Identify disconnected tree circuit(s) which is (are) caused by repeated link(s) and/or loop(s).

(v) Identify a repeated link and then erase this repeated link. Connect one of the nodes, which originally connected the repeated link, to the disconnected tree circuit. Thereby the disconnected tree circuit can be connected to the piping circuit.

(vi) If there is no repeated link in (v), identify a loop and then erase one of the links in the loop. Connect one of the nodes, which originally connected the link of the loop, to the disconnected tree circuit. Thereby the disconnected tree circuit can be connected to the piping circuit.

Table 6.4 shows the experimental results for a problem of size 17 with penalty value and repairing, respectively. A static penalty function method with value of 30,000 x 10^6 was used in this study. This is the function value of the worst piping configuration found in the GA optimisation process. It can be seen from Table 6.4 that both cases can find the optimal solution (2,913 x 10^6), but that the success rate increased from 35% to 60% with the incorporation of a repairing algorithm applied to infeasible candidates. Therefore more feasible candidates can be involved in the GA optimisation process and hence the performance improved. Moreover, in the case with repairing, the average best function value is 2,954 x 10^6 which is 6.5% better than the case with penalty value.

Table 6.4
Experimental results over 20 runs for problem of size 17 with penalty value and repairing

	Looped Local Search with Penalty Value		Looped Local Search with Repairing	
Run no.	Best Function Value ($x10^6$)	First Hit Generation No.	Best Function Value ($x10^6$)	First Hit Generation No.
1	3,305	446	2,913	768
2	3,225	871	2,913	628
3	3,599	334	3,006	528
4	2,913	395	2,913	216
5	3,395	259	3,034	263
6	2,913	897	2,994	248
7	3,080	460	2,913	391
8	2,913	866	2,913	761
9	3,178	776	2,924	231
10	3,131	310	2,913	472
11	2,913	498	2,994	349
12	3,225	725	2,913	458
13	2,913	662	3,101	685
14	2,913	595	3,006	293
15	3,178	398	2,913	655
16	3,599	784	2,913	531
17	3,599	558	2,913	491
18	2,913	698	3,070	316
19	3,225	368	2,913	433
20	3,080	411	2,913	582
Min.	2,913	259	2,913	216
Average	3,161	566	2,954	456
Std Dev	241.5	205.4	61.2	176.2
Succ R	35%	-	60%	-

6.4 Candidate Selection for Looped Local Search

During the GA optimisation process, candidates are selected from the population to undergo Local Search/Looped Local Search. Obviously, the elite (i.e. the best among the populations) was selected in order to search for a better solution in its vicinity.

However, there is a risk that the optimisation process traps itself into another local optimum. Randomly selecting a candidate from the population, without regard to its function value, may be another effective approach. In Table 6.5, a comparison of the performance of GA with Looped Local Search between these two different selection approaches is reported.

The results show that the case with the elite as candidate for Looped Local Search has a better performance than the one with random candidate-selection approach in both the average best function value and success rate. Moreover, the latter could not find the best solution ($2{,}913 \times 10^6$). Therefore, the elite candidate-selection method was kept for implementation in the present study.

Table 6.5
Experimental results over 20 runs for problem of size 17 with 2 different candidate selection approaches

	Looped Local Search (LLS) with Elite as Candidate for LLS		Looped Local Search (LLS) with Randomly Selected Candidate for LLS	
Run no.	Best Function Value ($x10^6$)	First Hit Generation No.	Best Function Value ($x10^6$)	First Hit Generation No.
1	2,913	768	3,764	673
2	2,913	628	3,193	456
3	3,006	528	3,228	916
4	2,913	216	3,183	527
5	3,034	263	3,499	73
6	2,994	248	3,228	341
7	2,913	391	3,247	525
8	2,913	761	3,188	437
9	2,924	231	3,250	625
10	2,913	472	3,270	423
11	2,994	349	3,414	482
12	2,913	458	3,183	689
13	3,101	685	3,183	598
14	3,006	293	3,624	844
15	2,913	655	3,553	795
16	2,913	531	3,006	728
17	2,913	491	3,250	595
18	3,070	316	3,183	936
19	2,913	433	3,101	687
20	2,913	582	3,247	774
Min.	2,913	216	3,006	73
Average	2,954	456	3,290	606
Std Dev	61.2	176.2	186.6	208.5
Succ R	60%	-	0%	-

6.5 Three Types of Looped Local Search

In the Looped Local Search performed in Sections 6.2 to 6.4, the length of the piping segment is used as a selection criterion to identify the link to undergo the Looped Local

Search process. It is known that the pipe diameter and the links along the critical path are also potential criteria for selecting suitable links during the local search process since they are also crucial factors affecting the function value (Chan *et al.*, 2007b). In order to have a full understanding on this area, a Looped Local Search using "minimum pipe diameter" and "longest link along the critical path" as link selection criteria have been incorporated into this study.

For the case using minimum pipe diameter as the link selection criterion, similar procedures as that for "longest links" were used (see Section 5.3). The $link_{k,\ longest\ length}$ is replaced by $link_{k,\ min.\ dia.}$. The detailed procedures of the "minimum pipe diameter" selection criterion used in Local Search/Looped Local Search are listed below:

Local Search

(i) If the function value converges and there is no further improvement by the GA operators (crossover & mutation) after x generations (arbitrarily assigned), subroutine "Local Search/Looped Local Search" will be called.

(ii) The elite in the current generation will be taken as the candidate for Local Search/Looped Local Search.

(iii) The link with minimum pipe diameter ($link_{min.\ dia.}$) of the elite and the two nodes ($node_{NB}$ and $node_{NE}$) joining this link are identified.

(iv) Take $node_{NB}$ as a searching point and be connected to the other $node_j$, ($j \in \{1, 2, ..., m\}$, $j \neq$ NB or NE), to form a new $link_{NB\ -\ j}$.

(v) The new link: $link_{NB\ -\ j}$ replaces the original link with minimum pipe diameter, $link_{min.\ dia.}$, to form a new piping network.

(vi) Calculate the function value of this new network.

If function $value_{NB-j} \geq$ function $value_{elite}$, do nothing.

If function $value_{NB-j} <$ function $value_{elite}$, assign a new best function value: function $value_{elite} \leftarrow$ function $value_{NB-j}$ and elite $\leftarrow$ new solution.

(vii) Repeat steps (iv) – (vi) using other nodes of j ($j \in \{1, 2, ..., m\}$, $j \neq$ NB or NE) until j = m.

(viii) Repeat steps (iv) – (vii) using $node_{NE}$ as the searching point.

Looped Local Search

(ix) If there is no further improvement by GA operators (crossover & mutation) and Local Search after x generations, the link with the second minimum pipe diameter ($link_{k,\ min.\ dia.}$) $_{(k=2^{nd})}$ of the elite and the two nodes ($node_{NB,k}$ and $node_{NE,k}$) joining the link are identified.

(x) Take $node_{NB,\ k}$ as a searching point and be connected to the other $node_j$, ($j \in \{1, 2, ..., m\}$, $j \neq NB_{,k}$ or $NE_{,k}$), to form a new $link_{NB,k-j}$.

(xi) The new link: $link_{NB,k-j}$ replaces the original link, $link_{k,\ min.\ dia}$, to form a new piping network.

(xii) Calculate the function value of this new network.

If function $value_{NB,k-j} \geq$ function $value_{elite}$, do nothing.

If function $value_{NB,k-j} <$ function $value_{elite}$, assign a new best function value: function $value_{elite} \leftarrow$ function $value_{NB,k-j}$ and elite $\leftarrow$ new solution.

(xiii) Repeat steps (x) – (xii) using other nodes of j ($j \in \{1, 2, ..., m\}$, $j \neq NB_{,k}$ or $NE_{,k}$) until j = m.

(xiv) Repeat steps (x) – (xiii) using $node_{NE,k}$ as the searching point.

(xv) Repeat steps (ix) – (xiv) for k = (3^{rd}, 4^{th}, ……., $m - 1^{th}$), i.e. the link with the third minimum pipe diameter, the link with the fourth minimum pipe diameter, ……,

the link with the $m - 1^{th}$ minimum pipe diameter.

For the case with the links along the critical path as link selection criterion, the searching procedures of Local Search/Looped Local Search as outlined in Chapter 5 (Section 5.3) were followed, except that only the links along the critical path were selected for implementing the Local Search/Looped Local Search.

Three sets of parametric experiments have been run with pipe length, pipe diameter and critical path as the link selection criteria for Looped Local Search. The results are shown in Table 6.6

From the results shown in Table 6.6, it is found that both the cases with "longest length" and "minimum diameter" as link selection criteria can find the optimal solution of 2,913 x 10^6 while the case with "critical path" fails. The case with length-criterion is still the best approach as the success rate is high (60%) while the case with diameter-criterion is only 15%. The case with "critical path" performs worst in terms of both solution quality and computational efficiency. Therefore, using the "longest length" as the link selection criterion appears to be the best approach in implementing Local Search/Looped Local Search in this study.

Table 6.6
Experimental results over 20 runs for problem of size 17 with three types of link selection criteria

Run no.	Longest Length		Minimum Diameter		Critical Path	
	Best Function Value ($x10^6$)	First Hit Generation No.	Best Function Value ($x10^6$)	First Hit Generation No.	Best Function Value ($x10^6$)	First Hit Generation No.
1	2,913	768	3,010	1,000	3,345	845
2	2,913	628	3,279	699	3,345	845
3	3,006	528	2,937	683	3,435	906
4	2,913	216	2,999	970	3,608	984
5	3,034	263	3,200	991	3,202	966
6	2,994	248	3,144	660	3,507	992
7	2,913	391	2,913	975	3,245	946
8	2,913	761	2,991	983	3,726	970
9	2,924	231	2,977	957	4,717	752
10	2,913	472	3,514	654	3,198	882
11	2,994	349	2,995	748	3,379	798
12	2,913	458	2,969	966	3,544	825
13	3,101	685	2,913	901	3,250	816
14	3,006	293	3,963	472	3,202	924
15	2,913	655	2,962	916	3,248	769
16	2,913	531	2,913	692	3,539	885
17	2,913	491	3,431	545	3,578	932
18	3,070	316	3,006	881	3,726	812
19	2,913	433	3,289	959	3,345	762
20	2,913	582	2,918	877	3,260	776
Min.	2,913	216	2,913	472	3,198	752
Average	2,954	456	3,116	826	3,470	869
Std Dev	61.2	176.2	268.2	165.6	339.9	79.7
Succ R	60%	-	15%	-	0%	-

6.6 Frequency of Local Search

In this study, local search was not applied for the selected candidate in every generation, since this is time consuming and may not be effective. Therefore a local search was only applied if a local optimum is found, i.e., there is no further improvement for the value of the objective function after a certain number of generations. Initially, a frequency of local search (FoLS) of 10 was used. FoLS is the number of generations after which there is no improvement in the best function value and then Local Search/Looped Local Search will be called and implemented. In order to investigate the effect of different values of FoLS on the GA search performance, two values of FoLS (>5 and >10) are applied to the cases of problem size 17, with Local Search and Looped Local Search. The experimental results over 20 runs are shown in Tables 6.7 and 6.8 respectively.

The experiment with FoLS>10 is taken as a base case. As seen from Table 6.7, for the case with Local Search and FoLS>5, a better best function value of 3,186 x 10^6 was found. However, in terms of average best function, it gave a higher value of 3,634 x 10^6. For the cases with Looped Local Search, the base case outperforms in both average best function value and success rate. The values are 2,954 x 10^6 and 60%, respectively while that for the case with FoLS>5 are 3,134 x 10^6 and 40%. On the other hand, the base case produced a higher average first hit generation number (456). There is no clear conclusion that smaller or larger FoLS can give better performance of GA with Local Search/Looped Local Search. Therefore, the original FoLS>10 was kept for the present study.

Table 6.7
Experimental results over 20 runs for problem of size 17 with Local Search at different frequency of local search (FoLS)

Run no.	With Local Search FoLS > 10		With Local Search FoLS > 5	
	Best Function Value ($x10^6$)	First Hit Generation No.	Best Function Value ($x10^6$)	First Hit Generation No.
1	3,698	418	3,698	418
2	4,100	717	3,804	695
3	4,033	434	3,186	634
4	3,547	766	4,641	970
5	3,360	960	3,507	839
6	3,257	648	3,740	380
7	3,406	602	3,780	916
8	3,392	785	3,446	679
9	3,214	769	3,464	918
10	3,360	990	3,350	740
11	3,406	698	3,780	559
12	3,406	749	3,446	472
13	4,204	954	3,450	613
14	3,698	825	3,977	646
15	3,547	841	3,536	825
16	3,214	592	3,873	578
17	3,656	681	3,464	798
18	3,392	636	3,581	885
19	3,214	499	3,464	984
20	3,592	587	3,490	863
Min.	3,214	418	3,186	380
Average	3,535	708	3,634	721
Std Dev	292.4	162.5	307.6	182.1
Succ R	0%	-	15%	-

Table 6.8
Experimental results over 20 runs for problem of size 17 with Looped Local Search at different frequency of local search (FoLS)

	With Looped Local Search FoLS > 10		With Looped Local Search FoLS > 5	
Run no.	Best Function Value ($x10^6$)	First Hit Generation No.	Best Function Value ($x10^6$)	First Hit Generation No.
1	2,913	768	3,281	193
2	2,913	628	2,913	722
3	3,006	528	3,049	330
4	2,913	216	2,913	508
5	3,034	263	3,224	145
6	2,994	248	3,482	471
7	2,913	391	3,110	192
8	2,913	761	3,268	843
9	2,924	231	2,913	279
10	2,913	472	3,203	206
11	2,994	349	2,913	484
12	2,913	458	2,913	359
13	3,101	685	3,370	480
14	3,006	293	2,913	625
15	2,913	655	3,341	609
16	2,913	531	2,913	517
17	2,913	491	3,260	363
18	3,070	316	3,405	509
19	2,913	433	2,913	361
20	2,913	582	3,384	425
Min.	2,913	216	2,913	145
Average	2,954	456	3,134	431
Std Dev	61.2	176.2	208.1	184.1
Succ R	60%	-	40%	-

6.7 When to Start Local Search

Another interesting point is when to start the Local Search/Looped Local Search. The performance of a GA may be more efficient if the local search is only called after the initial stage in which the solution is searched solely by the genetic operators (crossover

and mutation). In this section, each run was for 1,000 generations divided into two halves. In the first 300 generations, only genetic operators were used in the optimisation process. Then Looped Local Search was called after the first 300 generations. The results are shown in Table 6.9.

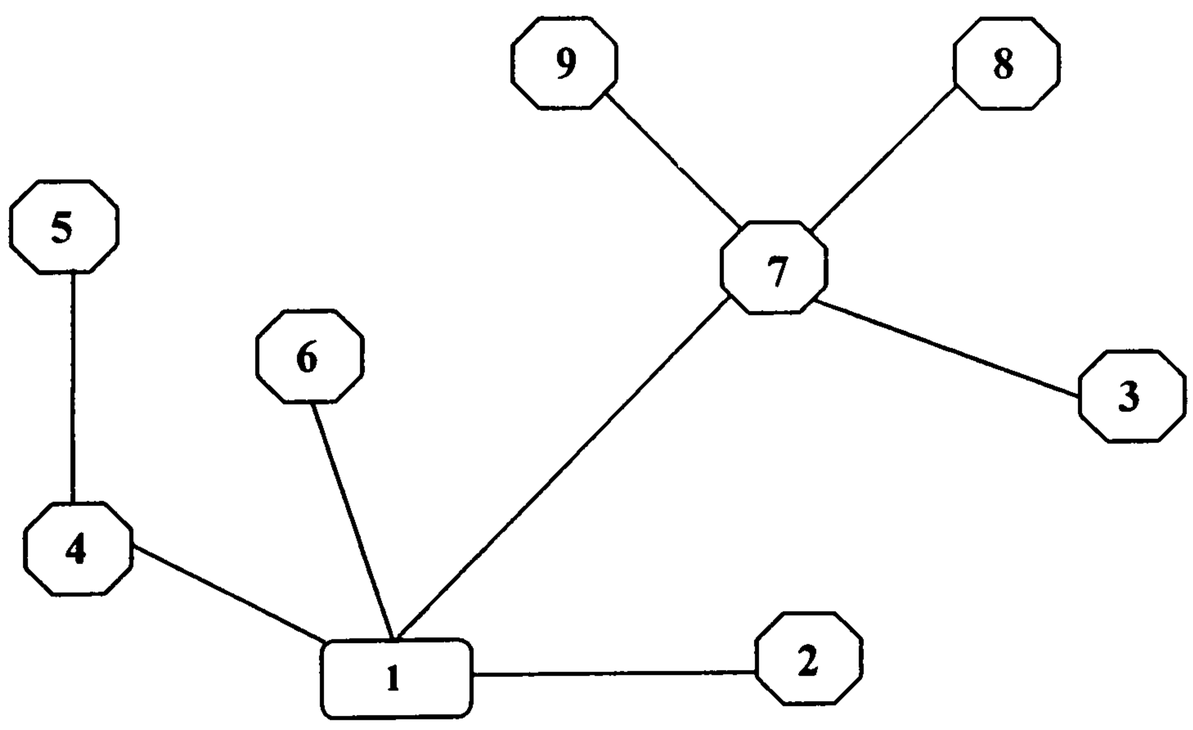
Table 6.9
Experimental results over 20 runs for problem of size 17 with Looped Local Search starting at different generation numbers

	With Looped Local Search Being Called after Generation No.1		With Looped Local Search Being Called after Generation No.300	
Run no.	Best Function Value ($x10^6$)	First Hit Generation No.	Best Function Value ($x10^6$)	First Hit Generation No.
1	2,913	768	3,249	513
2	2,913	628	3,019	617
3	3,006	528	3,003	798
4	2,913	216	3,018	883
5	3,034	263	3,079	443
6	2,994	248	2,913	715
7	2,913	391	3,068	692
8	2,913	761	2,913	778
9	2,924	231	3,092	817
10	2,913	472	3,097	492
11	2,994	349	3,456	530
12	2,913	458	2,913	836
13	3,101	685	2,950	889
14	3,006	293	3,037	539
15	2,913	655	2,932	825
16	2,913	531	3,066	787
17	2,913	491	3,360	489
18	3,070	316	3,272	598
19	2,913	433	3,046	745
20	2,913	582	2,937	890
Min.	2,913	216	2,913	443
Average	2,954	456	3,071	694
Std Dev	61.2	176.2	152.8	152.0
Succ R	60%	-	15%	-

It can be seen that both approaches can find the best function value (2,913 x 10^6). However, in terms of success rate and average best function value, the case with Looped Local Search being called after generation no. 1 performs better. The values are 60% and 2,954 x 10^6, respectively. There is no evidence that starting the Looped Local Search after a certain generation number can result better GA performance. Therefore, the original approach is recommended.

6.8 Concluding Remarks

Based on the experimental results obtained and presented in this chapter, the layouts of the best piping configurations for the DCS problems of sizes 9 and 17 are shown in Figures 6.3 and 6.4. Each edge in the piping network represents two parallel pipes, including the supply and return pipes.



(DCS central plant)

Figure 6.3 Optimal piping configuration for problem of size 9

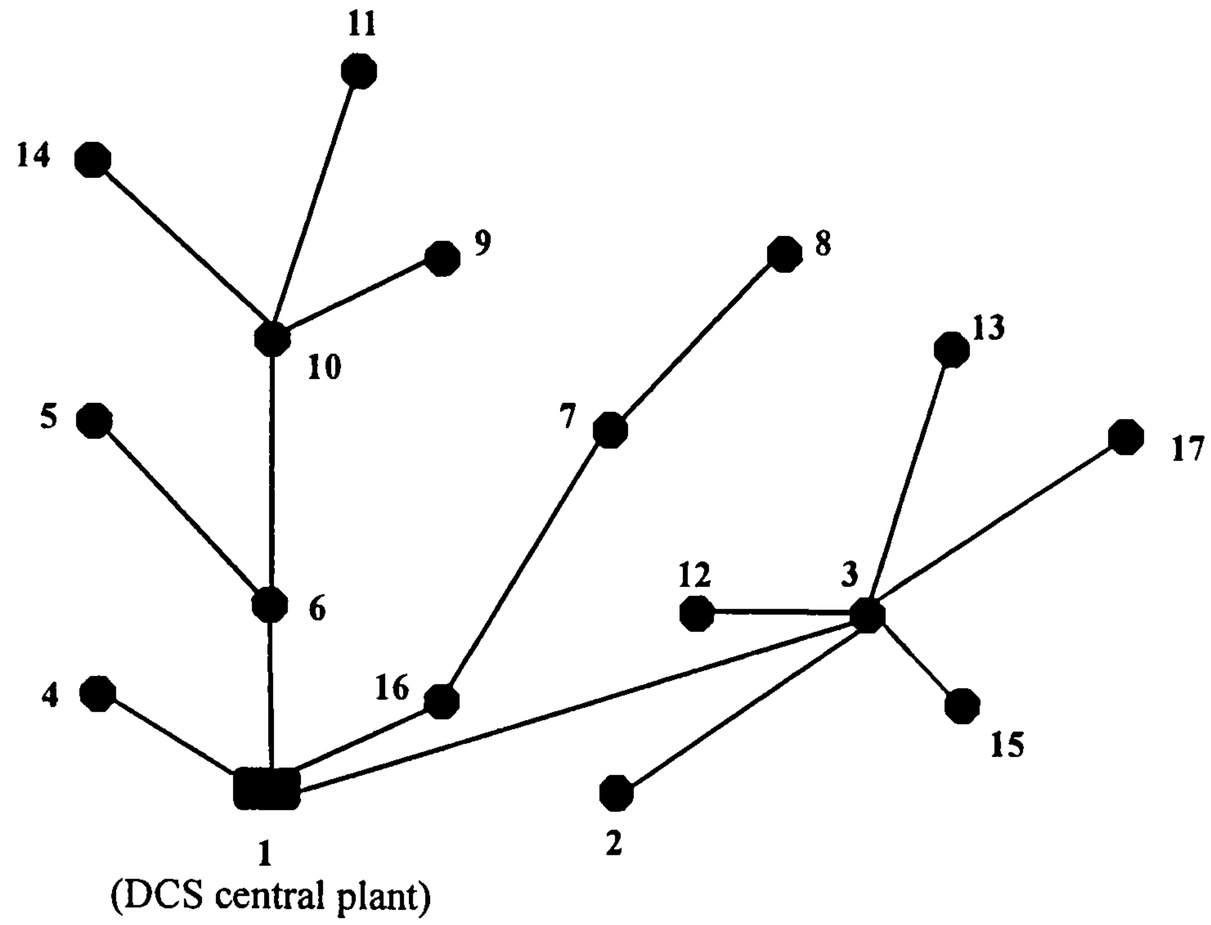


Figure 6.4 Optimal piping configuration for problem of size 17

In this chapter, the effects on GA performance of a number of factors have been investigated. The factors include:

- the effectiveness of Local Search and Looped Local Search implemented in the GA optimisation process;
- the application of a penalty value and repair algorithm for infeasible candidates;
- using an elite or randomly selected individual as a candidate to undergo Local Search/Looped Local Search;
- link selection criteria including "longest length", "minimum diameter" and "critical path";
- appropriate frequency of local search (FoLS);
- appropriate stage to start local search.

It was found that Looped Local Search (LLS) with repair algorithm at a FoLS > 10, using the elite candidate for implementing LLS, and applying "longest length" as the link selection criterion for the elite candidate is the optimal combination of settings to be used in a GA for optimisation of the distribution piping configuration in DCS. In order to validate the effectiveness of the developed algorithm with optimal settings, benchmark problems have also been used for comparison and the details are presented in the next chapter.

CHAPTER 7
COMPARISON WITH BENCHMARK PROBLEMS

In the previous chapter, the effect of a number of factors on GA performance for the pipe work optimisation problem was investigated through parametric experiments. In order to validate its effectiveness, a benchmark problem – the optimal communication spanning tree problem was chosen for comparison. In the following sections, the definition and features of this type of benchmark problem are introduced. Then the developed algorithm is applied to this benchmark problem to illustrate the superior performance of GA with Local Search/Looped Local Search technique.

7.1 Optimal Communication Spanning Tree Problem

In the optimal communication spanning tree (OCST) problem, the objective is to find a network to connect all the given nodes (demand points) and satisfy their communication requirements at a minimum total cost (Rothlauf, 2002). For the OCST problem, the number and positions of the nodes in a network are pre-fixed. The cost of the network is determined by the cost of the links. The flow in a link is the sum of the communication demands between all pairs of nodes communicating either directly or indirectly over the link. The cost for each link is not pre-set but depends on its distance weight and its capacity. The capacity of a link must satisfy the flow over the link which depends on the entire network structure. Since the features of the OCST problem are very similar to that of the DCS piping network in the present study, a benchmark problem of this type has been selected to test the effectiveness of the developed Local Search/Looped Local Search algorithm.

The OCST problem can be defined as an undirected graph which is denoted as $G = (V, E)$. The number of nodes is denoted as $n = |V|$ and the number of edges is denoted as $|E|$. There are communication demands between the n different nodes. The demands are represented by a $n \times n$ demand matrix $R = (r_{ij})$, where r_{ij} is the amount of traffic required between location v_i and v_j. A $n \times n$ distance matrix $D = d_{ij}$ determines the distance weights associated with each pair of sites. A tree $T = (V, F)$ where $F \subseteq E$ and $|F| = |V| - 1$ is called a spanning tree of G if it connects all the nodes. The weight $c(T)$ of the spanning tree is the weighted sum over all pairs of nodes of the cost of the path between all pairs in T.

$$c(T) = \sum_{i,j \in F} f(d_{ij}, b_{ij}) \tag{7.1}$$

where the $n \times n$ matrix $B = b_{ij}$ denotes the traffic flowing directly and indirectly over the link between the nodes i and j.

For the OCST problem, the cost of a link is calculated as the product of the distance weight d_{ij} times the overall traffic b_{ij} running over the link. Therefore, $f = d_{ij}\, b_{ij}$ and

$$c(T) = \sum_{i,j \in V, i<j} r_{ij} \times d(p_{i,j}^{T}) \tag{7.2}$$

where $d(p_{i,j}^{T})$ denotes the weight of the unique path from node i to node j in the spanning tree. The OCST problem searches for the spanning tree with minimal costs among all other spanning trees.

7.2 Palmer's Test Problem

The first benchmark problem comes from Palmer's previous work on the OCST problem (Palmer, 1994). One of Palmer's OCST problems with 12 nodes was selected for the present study. The inter-node traffic requirements were inversely proportional to the distances between nodes. The nodes corresponded to cities in the United States and the distances between the nodes $d_{i,j}$ were obtained from a tariff database. The requirements were based on information including the population of each city, the distance between cities, and other factors, all of which had control parameters for determining the influence of each kind of information. The cost of each link was calculated as the traffic over each link multiplied by the distance weight of each link. An example of the communication network is shown in Figure 7.1. Tables 7.1 and 7.2 list the demands and distance weights for this Palmer's OCST problem. The best known solution and the links used in the corresponding best network configuration are shown in Table 7.3.

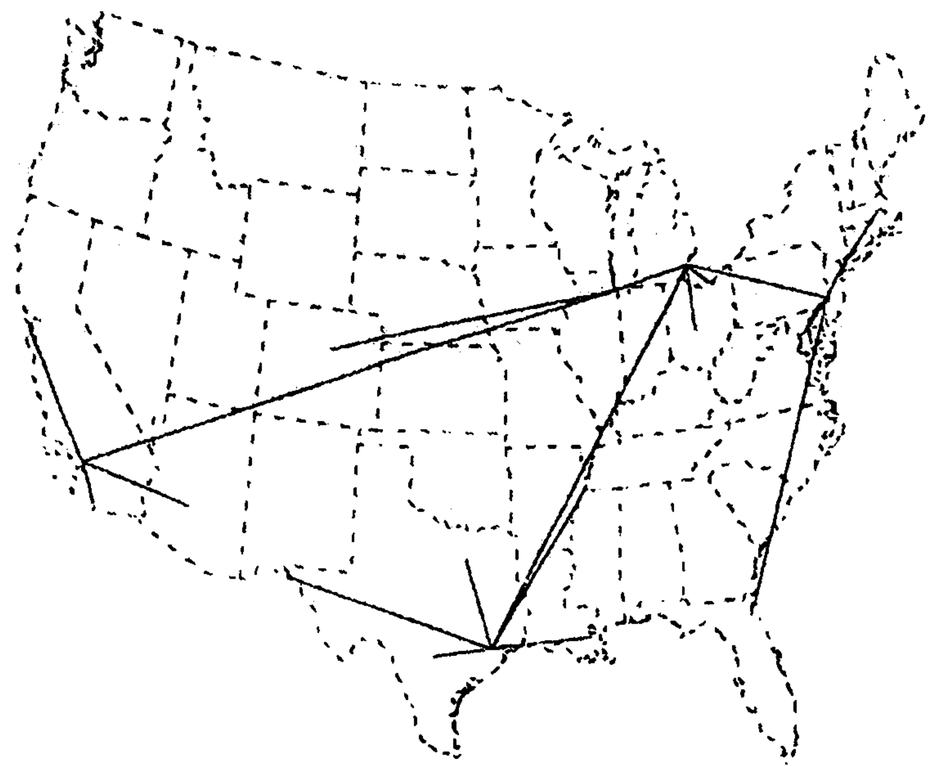


Figure 7.1 An example of communication network for Palmer's test problem (This figure is extracted from Palmer's thesis (Palmer, 1994))

Table 7.1
Demand matrix for Palmer's OCST problem

	PHNX	LA	SD	SF	CHI	BALT	DET	NY	PHIL	DAL	HOU	SANAN
PHNX	0	7	8	4	2	1	2	1	1	3	3	3
LA	-	0	25	7	1	1	1	1	1	2	2	2
SD	-	-	0	6	1	1	1	1	1	2	2	2
SF	-	-	-	0	1	1	1	1	1	2	2	2
CHI	-	-	-	-	0	4	11	4	4	3	3	2
BALT	-	-	-	-	-	0	6	15	29	2	2	2
DET	-	-	-	-	-	-	0	5	6	3	2	2
NY	-	-	-	-	-	-	-	0	33	2	2	2
PHIL	-	-	-	-	-	-	-	-	0	2	2	2
DAL	-	-	-	-	-	-	-	-	-	0	11	10
HOU	-	-	-	-	-	-	-	-	-	-	0	14
SANAN		-	-	-	-	-	-	-	-	-	-	0

Table 7.2
Distance matrix for Palmer's OCST problem

	PHNX	LA	SD	SF	CHI	BALT	DET	NY	PHIL	DAL	HOU	SANAN
PHNX	0	6,490	5,903	8,484	14,561	18,359	15,976	19,360	18,867	10,090	10,883	9,665
LA	-	0	4,523	6,256	16,661	20,618	18,083	21,561	21,099	12,639	13,461	12,236
SD	-	-	0	6,908	16,414	20,292	17,829	21,263	20,787	12,073	12,802	11,540
SF	-	-	-	0	17,328	21,452	18,714	22,286	21,874	14,234	15,259	14,136
CHI	-	-	-	-	0	8,425	5,658	9,194	8,797	9,603	10,440	11,237
BALT	-	-	-	-	-	0	6,621	5,067	4,439	12,385	12,526	13,722
DET	-	-	-	-	-	-	0	7,230	6,899	10,720	11,340	12,297
NY	-	-	-	-	-	-	-	0	4,300	13,531	13,730	14,912
PHIL	-	-	-	-	-	-	-	-	0	12,967	13,130	14,319
DAL	-	-	-	-	-	-	-	-	-	0	4,888	5,076
HOU	-	-	-	-	-	-	-	-	-	-	0	4,478
SANAN		-	-	-	-	-	-	-	-	-	-	0

Table 7.3
Best known solution and links used for Palmer's test problem

	Best known solution $c(T_{best,f})$	Links used for the best known solution
Palmer's test problem (12 nodes)	3,428,509	SD-PHNX, SD-LA, SF-SD, DET-CHI, PHIL-BALT, PHIL-DET, PHIL-NY, DAL-SD, DAL-DET, HOU-DAL, SANAN-DAL

7.3 Raidl's Test Problem

In Raidl's OCST problem (Rothlauf, 2006), the distance weight $d_{i,j}$ and the traffic demands $r_{i,j}$ have been generated randomly and are uniformly distributed. The cost of a tree is calculated in the same way as that in Palmer's OCST problem, i.e. the cost of a link is calculated as the amount of traffic over a link multiplied by its distance weight. The demand and distance matrixes are summarised in Tables 7.4 and 7.5 respectively. The best known solution and the links used in the corresponding best network configuration are shown in Table 7.6.

Table 7.4
Demand matrix for Raidl's OCST problem

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	19	7	97	99	22	65	82	53	76	2	44	7	40	67	100	7	40	94	90
1	-	0	37	18	98	75	99	90	42	51	4	91	76	91	10	49	53	75	72	17
2	-	-	0	56	91	59	24	34	33	30	0	32	38	6	25	94	43	9	57	18
3	-	-	-	0	8	13	26	25	17	16	67	74	93	16	26	33	54	10	90	44
4	-	-	-	-	0	69	80	44	1	10	10	100	14	16	92	7	26	0	30	44
5	-	-	-	-	-	0	75	43	36	66	26	18	33	100	11	15	26	44	69	2
6	-	-	-	-	-	-	0	100	79	37	80	22	39	56	32	4	70	48	96	77
7	-	-	-	-	-	-	-	0	74	63	73	84	3	16	86	70	8	4	2	8
8	-	-	-	-	-	-	-	-	0	82	84	0	92	52	2	58	30	39	3	18
9	-	-	-	-	-	-	-	-	-	0	44	59	50	15	28	64	77	71	4	5
10	-	-	-	-	-	-	-	-	-	-	0	43	88	9	25	40	79	34	44	47
11	-	-	-	-	-	-	-	-	-	-	-	0	8	92	30	8	83	82	77	40
12	-	-	-	-	-	-	-	-	-	-	-	-	0	78	82	43	96	93	68	11
13	-	-	-	-	-	-	-	-	-	-	-	-	-	0	7	96	75	84	66	79
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	90	76	33	99	0
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	73	43	0	83
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	90	8	74
17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	86	83
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	22
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Table 7.5
Distance matrix for Raidl's OCST problem

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	30	6	15	67	100	99	34	85	56	3	13	23	2	72	24	53	12	94	57
1	-	0	98	30	53	35	38	59	85	82	85	78	16	3	59	73	17	77	73	15
2	-	-	0	62	9	70	65	21	44	18	44	68	71	56	13	79	5	43	83	39
3	-	-	-	0	80	93	23	61	78	52	28	80	62	1	48	38	25	62	100	33
4	-	-	-	-	0	84	16	6	27	85	49	46	7	4	59	37	8	53	19	98
5	-	-	-	-	-	0	32	12	20	92	41	71	20	72	32	72	19	22	96	80
6	-	-	-	-	-	-	0	73	80	15	88	85	94	72	34	39	79	89	49	15
7	-	-	-	-	-	-	-	0	49	1	86	46	32	97	66	76	37	88	47	8
8	-	-	-	-	-	-	-	-	0	72	18	78	93	65	57	65	44	24	4	29
9	-	-	-	-	-	-	-	-	-	0	17	75	14	55	5	54	56	72	2	56
10	-	-	-	-	-	-	-	-	-	-	0	57	100	40	5	17	67	93	4	13
11	-	-	-	-	-	-	-	-	-	-	-	0	52	75	82	29	19	46	37	83
12	-	-	-	-	-	-	-	-	-	-	-	-	0	41	60	38	21	76	13	86
13	-	-	-	-	-	-	-	-	-	-	-	-	-	0	83	69	40	90	40	93
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	97	48	92	36	52
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	8	2	44	12
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	66	95	38
17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	99	23
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	57
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Table 7.6
Best known solution and links used for Raidl's test problem

	Best known solution $c(T_{best,f})$	Links used for the best known solution
Raidl's test problem (20 nodes)	157,570	2-0, 7-5, 9-6, 9-7, 10-0, 11-0, 12-4, 13-0, 13-1, 13-3, 13-4, 14-10, 16-2, 17-0, 17-15, 18-8, 18-9, 18-10, 19-10

7.4 Berry's Test Problem

Berry *et al.* (1995) proposed three instances of the OCST problems which consisted of one 6-node and two 35-node problems. Both 35 node problems used the same traffic demands, but differed in the distance weights. One problem had uniform distance weights with $d_{ij} = 1$, and the other one had non-uniform distance weights. In the present study, the one with 35 nodes and non-uniform distance weights was selected. The corresponding demand and distance matrixes are listed in Tables 7.7 and 7.8. The best known solution and the links used in the corresponding best network configuration are shown in Table 7.9.

Table 7.7
Demand matrix for Berry's OCST problem

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
0	0	639	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	93	0	0	0	0	0	0	0	129	0
1	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	147	0	0	0	0	0	0	0	83	0	0
2	-	-	0	0	0	0	0	0	189	0	0	0	0	0	0	0	99	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	43	0	0
3	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	531	0	0	0	0	0	0	0	623	0	0	0	0	0	0	0	0	0
4	-	-	-	-	0	0	0	0	0	0	0	53	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	-	-	-	-	-	0	43	0	0	0	0	0	0	0	119	0	0	0	0	0	0	0	329	0	0	0	0	0	0	0	651	0	0	0	0
6	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	371	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	171	0	0	0
8	-	-	-	-	-	-	-	-	0	0	0	0	0	41	0	0	0	0	0	0	0	189	0	0	0	0	0	0	0	123	0	0	0	0	0
9	-	-	-	-	-	-	-	-	-	0	0	0	351	0	0	0	0	0	0	0	61	0	0	0	0	0	0	0	217	0	0	0	0	0	0
10	-	-	-	-	-	-	-	-	-	-	0	0	81	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	133	0	0	0	0	0
12	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	27	0	0	0	0	0	0	0	161	0	0	0	0	0	0	0	261	0	0	0
13	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	261	0	0	0	0	0	0	0	639	0	0	0	0
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	147	0	0	0	0	0	0	0	423	0	0	0	0	0	0	0
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	69	0	0	0	0	0	0	0	0	0
17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	351	0	0	0	0	0	0	0	117	0	0
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	243	0	0	0	0	0	0	0	873	0	0
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	639	0	0	0	0	0	0	0	119	0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	57	0	0	0	0	0	0	0
21	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	91	0	0	0	0	0	0	0	387	0	0	0	0
22	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	89	0	0	0	0	0	0	0	0
23	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	651	0	0	0
24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	133	0	0	0	0	0
25	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	21	0	0	0	0	0	0
26	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0
27	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	111	0	0	0	0	0
28	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
29	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	63
30	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0
31	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
32	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	71	0
33	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0
34	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

(Table 7.7 Cont'd)

Table 7.8
Distance matrix for Berry's OCST problem

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
0	0	1	7	4	8	9	4	12	6	9	11	7	10	7	9	7	8	3	3	6	8	7	10	12	2	5	9	8	10	6	8	11	4	3	7
1	-	0	6	3	7	8	3	11	5	8	10	6	9	6	8	6	7	2	2	5	7	6	9	11	1	4	8	7	9	5	7	10	3	2	6
2	-	-	0	3	5	4	7	9	1	6	8	4	7	2	4	4	1	4	6	3	5	2	5	9	5	2	4	5	7	3	3	8	7	6	4
3	-	-	-	0	4	5	4	8	2	5	7	3	6	3	5	3	4	1	3	2	4	3	6	8	2	1	5	4	6	2	4	7	4	3	3
4	-	-	-	-	0	7	8	10	4	7	9	1	8	5	7	5	6	5	7	4	6	5	8	10	6	3	7	6	8	2	6	9	8	7	3
5	-	-	-	-	-	0	9	11	3	8	10	6	9	4	2	6	5	6	8	5	7	2	3	11	7	4	2	7	9	5	1	10	9	8	6
6	-	-	-	-	-	-	0	12	6	9	11	7	10	7	9	7	8	3	1	6	8	7	10	12	2	5	9	8	10	6	8	11	2	3	7
7	-	-	-	-	-	-	-	0	8	3	3	9	2	9	11	5	10	9	11	6	4	9	12	2	10	7	11	6	4	8	10	1	12	11	9
8	-	-	-	-	-	-	-	-	0	5	7	3	6	1	3	3	2	3	5	2	4	1	4	8	4	1	3	4	6	2	2	7	6	5	3
9	-	-	-	-	-	-	-	-	-	0	2	6	1	6	8	2	7	6	8	3	1	6	9	3	7	4	8	3	1	5	7	2	9	8	6
10	-	-	-	-	-	-	-	-	-	-	0	8	1	8	10	4	9	8	10	5	3	8	11	3	9	6	10	5	3	7	9	2	11	10	8
11	-	-	-	-	-	-	-	-	-	-	-	0	7	4	6	4	5	4	6	3	5	4	7	9	5	2	6	5	7	1	5	8	7	6	2
12	-	-	-	-	-	-	-	-	-	-	-	-	0	7	9	3	8	7	9	4	2	7	10	2	8	5	9	4	2	6	8	1	10	9	7
13	-	-	-	-	-	-	-	-	-	-	-	-	-	0	4	4	3	4	6	3	5	2	5	9	5	2	4	5	7	3	3	8	7	6	4
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	6	5	6	8	5	7	2	3	11	7	4	2	7	9	5	1	10	9	8	6
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	5	4	6	1	1	4	7	5	5	2	6	1	3	3	5	4	7	6	4
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	5	7	4	6	3	6	10	6	3	5	6	8	4	4	9	8	7	5
17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	2	3	5	4	7	9	1	2	6	5	7	3	5	8	3	2	4
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	5	7	6	9	11	1	4	8	7	9	5	7	10	1	2	6
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	2	3	6	6	4	1	5	2	4	2	4	5	6	5	3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	5	8	4	6	3	7	2	2	4	6	3	8	7	5
21	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	3	9	5	2	2	5	7	3	1	8	7	6	4
22	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	12	8	5	1	8	10	6	2	11	10	9	7
23	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	10	7	11	6	4	8	10	1	12	11	9
24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	3	7	6	8	4	6	9	2	1	5
25	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	4	3	5	1	3	6	5	4	2
26	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	7	9	5	1	10	9	8	6
27	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	4	4	6	5	8	7	5
28	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	6	8	3	10	9	7
29	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	4	7	6	5	1
30	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	9	8	7	5
31	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	11	10	8
32	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	3	7
33	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	6
34	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

(Table 7.8 Cont'd)

Table 7.9
Best known solution and links used for Berry's test problem

	Best known solution $c(T_{best,f})$	Links used for the best known solution
Berry's test problem (35 nodes)	16,915	1-0, 8-2, 11-4, 12-9, 12-10, 13-8, 16-2, 17-3, 18-6, 19-15, 20-9, 20-15, 21-8, 24-1, 24-17, 24-18, 25-3, 25-8, 25-19, 26-22, 27-15, 28-9, 29-11, 29-25, 30-5, 30-14, 30-21, 30-22, 31-7, 31-12, 31-23, 32-18, 33-24, 34-29

7.5 Rothlauf's Real-World Test Problems

The communication network problems from Rothlauf (2006) were derived from a real-world, 26-node problem from a company with locations all over Germany. For fulfilling the demands between the nodes, different line types with discrete capacities and cost were available. The cost for installing a line consisted of a fixed and length-dependent share. Both of them depended on the capacity of the link. The costs were based on the tariffs of the German Telecom from 1996 and represented the amount of money (in German Marks) which a company needed to pay for renting a telecommunication line of a specific length and capacity per month.

In total, there are four OCST problems. In problem (1), a 16-node problem with traffic ending only at node 0 is used. There are one headquarter and fifteen branch offices. This is the original design problem. All the fifteen branch offices (nodes 1 to 15) communicate only with the headquarters (node 0). Possible line capacities are 64 kBit/s, 512 kBit/s, and 2,048 kBit/s. The complexity of this problem is low.

In the second problem, one of the nodes is removed and some additional traffic is added. There are one headquarter and only fourteen branches. Finding the best solution is slightly more involved than in problem (1).

The problem (3) is similar to problem (1) but modified cost functions for the lines are used. There are one headquarter and fifteen branches. In this problem, the fixed cost for installing a line is only 10% of the cost in problem (1). Therefore, the cost of a link is mainly determined by its distance weight. Hence, the optimal solution is more like a minimum spanning tree.

In the last OCST problem, there are traffic flows between all the sixteen nodes. There are four headquarters, twelve branches, and all are working together. The demand matrix is completely filled in this problem. Some traffic exists between every node *i* and *j*. Between the four headquarters (nodes 0, 1, 2 and 3) the traffic is uniformly distributed between 256 kBit/s and 512 kBit/s. Every other node communicates with the four headquarters and has a uniform demand between 0 and 512 kBit/s. The demand is split into the headquarters at a ratio of 0.4, 0.3, 0.2 and 0.1 for the nodes 0, 1, 2 and 3. Between all twelve branch offices, the demand of the traffic is uniformly distributed between 0 and 64 kBit/s. To make the problem more realistic, two additional line types are available. It is possible to use lines of 128 kBit/s and 4,096 kBit/s capacity with twice the cost of 64 kBit/s and 2,048 kBit/s lines.

The demand matrixes, distance matrixes and cost structures for the four Rothlauf's OCST problems are listed in Tables 7.10 to 7.17.

Table 7.10
Demand matrix for Rothlauf's OCST problems (1) and (3)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	424	458	727	468	414	440	521	50	48	381	34	28	48	34	28
1	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
4	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0
5	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0
6	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0
7	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0
8	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
9	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
10	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0
11	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0
12	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
13	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Table 7.11
Demand matrix for Rothlauf's OCST problem (2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	424	458	200	468	440	521	50	48	600	34	28	48	34	28
1	-	0	0	0	0	0	0	0	0	0	0	0	0	40	0
2	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0
3	-	-	-	0	0	0	0	0	0	0	0	0	0	0	0
4	-	-	-	-	0	0	0	0	0	0	0	0	0	0	100
5	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0
6	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0
7	-	-	-	-	-	-	-	0	0	0	0	0	0	0	0
8	-	-	-	-	-	-	-	-	0	0	0	0	0	0	0
9	-	-	-	-	-	-	-	-	-	0	0	0	0	0	0
10	-	-	-	-	-	-	-	-	-	-	0	0	0	0	0
11	-	-	-	-	-	-	-	-	-	-	-	0	0	0	0
12	-	-	-	-	-	-	-	-	-	-	-	-	0	0	0
13	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Table 7.12
Demand matrix for Rothlauf's OCST problem (4)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	308	491	364	36	51	195	72	114	111	14	150	78	136	33	44
1	-	0	503	323	27	38	146	54	86	83	11	112	59	102	24	33
2	-	-	0	272	18	25	97	36	57	55	7	75	39	68	16	22
3	-	-	-	0	9	12	48	18	28	27	3	37	19	34	8	11
4	-	-	-	-	0	51	17	1	34	40	54	36	47	45	25	11
5	-	-	-	-	-	0	15	63	22	16	31	42	28	54	33	7
6	-	-	-	-	-	-	0	5	26	62	54	45	39	12	16	18
7	-	-	-	-	-	-	-	0	32	13	40	22	20	34	61	38
8	-	-	-	-	-	-	-	-	0	35	16	54	13	38	49	17
9	-	-	-	-	-	-	-	-	-	0	10	12	47	4	5	49
10	-	-	-	-	-	-	-	-	-	-	0	49	10	55	28	39
11	-	-	-	-	-	-	-	-	-	-	-	0	10	4	48	37
12	-	-	-	-	-	-	-	-	-	-	-	-	0	19	41	38
13	-	-	-	-	-	-	-	-	-	-	-	-	-	0	17	34
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	36
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Table 7.13
Distance matrix for Rothlauf's OCST problems (1), (3) and (4)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	30.15	25.06	24.21	12.53	13.04	11.18	10.05	17.69	13.00	22.56	23.02	36.06	15.23	25.00	20.00
1	-	0	17.00	21.93	25.30	30.68	35.69	40.20	22.67	17.26	8.25	23.35	6.08	18.36	15.62	34.48
2	-	-	0	31.78	28.02	32.65	34.83	33.84	9.00	15.00	17.00	32.28	19.80	10.00	27.07	38.83
3	-	-	-	0	12.04	15.23	21.93	32.02	31.06	19.10	15.65	2.00	27.02	25.02	6.40	15.03
4	-	-	-	-	0	5.39	11.05	20.00	24.04	13.04	17.09	10.63	31.32	18.79	14.56	10.82
5	-	-	-	-	-	0	6.71	17.80	27.73	17.69	22.47	13.42	36.69	23.02	19.10	7.07
6	-	-	-	-	-	-	0	12.08	28.43	20.59	27.46	20.12	41.77	24.84	25.50	11.18
7	-	-	-	-	-	-	-	0	25.50	23.02	32.56	30.48	46.10	24.52	34.00	23.26
8	-	-	-	-	-	-	-	-	0	12.00	19.24	31.00	26.93	6.08	27.89	34.54
9	-	-	-	-	-	-	-	-	-	0	10.30	19.00	23.09	6.08	16.55	23.85
10	-	-	-	-	-	-	-	-	-	-	0	16.64	14.32	13.60	10.20	26.63
11	-	-	-	-	-	-	-	-	-	-	-	0	28.60	25.02	8.06	13.04
12	-	-	-	-	-	-	-	-	-	-	-	-	0	23.41	20.62	40.25
13	-	-	-	-	-	-	-	-	-	-	-	-	-	0	21.84	29.53
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	20.62
15	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Table 7.14

Distance matrix for Rothlauf's OCST problem (2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	30.15	25.06	24.21	12.53	11.18	10.05	17.69	13.00	22.56	23.02	36.06	15.23	25.00	20.00
1	-	0	17.00	21.93	25.30	35.69	40.20	22.67	17.26	8.25	23.35	6.08	18.36	15.62	34.48
2	-	-	0	31.78	28.02	34.83	33.84	9.00	15.00	17.00	32.28	19.80	10.00	27.07	38.83
3	-	-	-	0	12.04	21.93	32.02	31.06	19.10	15.65	2.00	27.02	25.02	6.40	15.03
4	-	-	-	-	0	11.05	20.00	24.04	13.04	17.09	10.63	31.32	18.79	14.56	10.82
5	-	-	-	-	-	0	12.08	28.43	20.59	27.46	20.12	41.77	24.84	25.50	11.18
6	-	-	-	-	-	-	0	25.50	23.02	32.56	30.48	46.10	24.52	34.00	23.26
7	-	-	-	-	-	-	-	0	12.00	19.24	31.00	26.93	6.08	27.89	34.54
8	-	-	-	-	-	-	-	-	0	10.30	19.00	23.09	6.08	16.55	23.85
9	-	-	-	-	-	-	-	-	-	0	16.64	14.32	13.60	10.20	26.63
10	-	-	-	-	-	-	-	-	-	-	0	28.60	25.02	8.06	13.04
11	-	-	-	-	-	-	-	-	-	-	-	0	23.41	20.62	40.25
12	-	-	-	-	-	-	-	-	-	-	-	-	0	21.84	29.53
13	-	-	-	-	-	-	-	-	-	-	-	-	-	0	20.62
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0

Table 7.15

Cost structure for Rothlauf's OCST problems (1) and (2)

Capacity b	Distance d	Cost C = f (d , b)
64 kBit/s	[0.00 ; 1.00]	334.58 d + 385.00
	]1.00 ; 3.00]*	148.70 d + 572.00
	]3.00 ; 10.00]	29.74 d + 972.50
	]10.00 ; ∞]	22.31 d + 1047.00
512 kBit/s	[0.00 ; 1.00]	1107.00 d + 975.00
	]1.00 ; 3.00]	520.00 d + 1567.00
	]3.00 ; 10.00]	178.00 d + 2717.00
	]10.00 ; ∞]	111.53 d + 3392.00
2048 kBit/s	[0.00 ; 1.00]	2215.00 d + 1950.00
	]1.00 ; 3.00]	1040.90 d + 3135.00
	]3.00 ; 10.00]	356.88 d + 5435.00
	]10.00 ; ∞]	223.05 d + 6785.00
> 2048 kBit/s	[0.00 ; ∞]	500000 d + 50000

*]1.00 ; 3.00] means ">1.00 and ≦ 3.00"

Table 7.16

Cost structure for Rothlauf's OCST problem (3)

Capacity b	Distance d	Cost C = f (d , b)
64 kBit/s	[0.00 ; 1.00]	334.58 d + 385.00
	]1.00 ; 3.00]	148.70 d + 572.00
	]3.00 ; 10.00]	29.74 d + 972.50
	]10.00 ; ∞]	22.31 d + 1047.00
512 kBit/s	[0.00 ; 1.00]	1107.00 d + 97.50
	]1.00 ; 3.00]	520.00 d + 156.70
	]3.00 ; 10.00]	178.00 d + 271.70
	]10.00 ; ∞]	111.53 d + 339.20
2048 kBit/s	[0.00 ; 1.00]	2215.00 d + 195.00
	]1.00 ; 3.00]	1040.90 d + 313.50
	]3.00 ; 10.00]	356.88 d + 543.50
	]10.00 ; ∞]	223.05 d + 678.50
> 2048 kBit/s	[0.00 ; ∞]	500000 d + 50000

Table 7.17

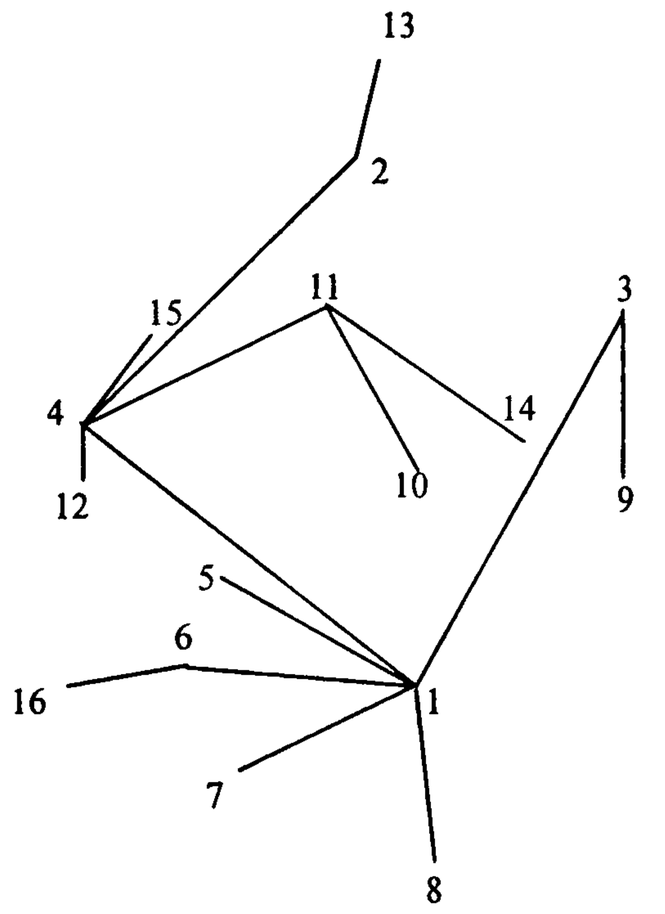
Cost structure for Rothlauf's OCST problem (4)

Capacity b	Distance d	Cost C = f (d , b)
64 kBit/s	[0.00 ; 1.00]	334.58 d + 385.00
	]1.00 ; 3.00]	148.70 d + 572.00
	]3.00 ; 10.00]	29.74 d + 972.50
	]10.00 ; ∞]	22.31 d + 1047.00
128 kBit/s	[0.00 ; 1.00]	669.16 d + 770.00
	]1.00 ; 3.00]	297.40 d + 1144.00
	]3.00 ; 10.00]	59.48 d + 1945.00
	]10.00 ; ∞]	44.62 d + 2094.00
512 kBit/s	[0.00 ; 1.00]	1107.00 d + 975.00
	]1.00 ; 3.00]	520.00 d + 1567.00
	]3.00 ; 10.00]	178.00 d + 2717.00
	]10.00 ; ∞]	111.53 d + 3392.00
2048 kBit/s	[0.00 ; 1.00]	2215.00 d + 1950.00
	]1.00 ; 3.00]	1040.90 d + 3135.00
	]3.00 ; 10.00]	356.88 d + 5435.00
	]10.00 ; ∞]	223.05 d + 6785.00
4096 kBit/s	[0.00 ; 1.00]	4430.00 d + 3900.00
	]1.00 ; 3.00]	2081.80 d + 6270.00
	]3.00 ; 10.00]	713.76 d + 10870.00
	]10.00 ; ∞]	446.10 d + 13570.00
> 2048 kBit/s	[0.00 ; ∞]	500000 d + 50000

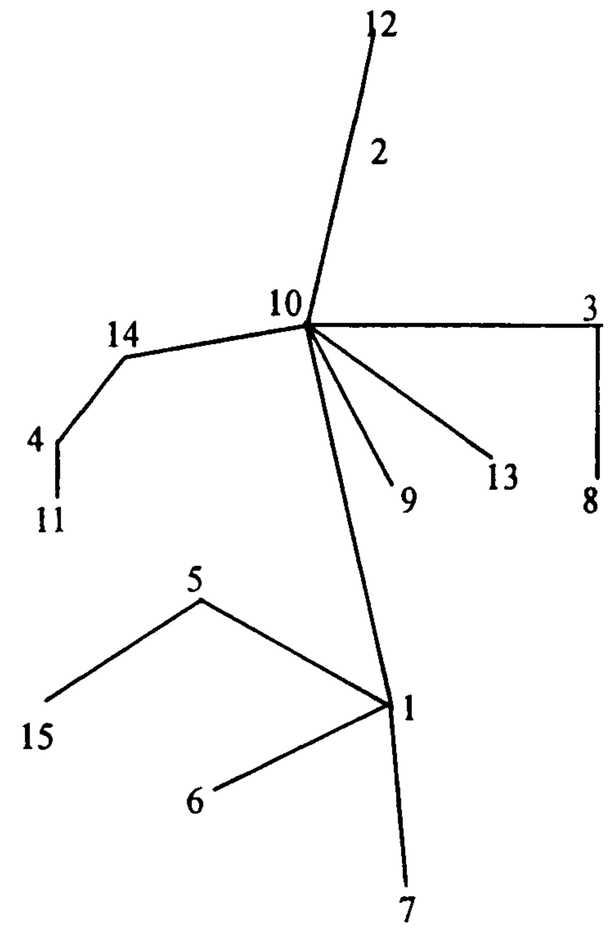
The best known solutions and the links used in the corresponding best network configurations for these four test problems are shown in Table 7.18. Moreover, the best known network configurations are illustrated in Figure 7.2.

Table 7.18
Best known solution and links used for Rothlauf's four real-world test problems

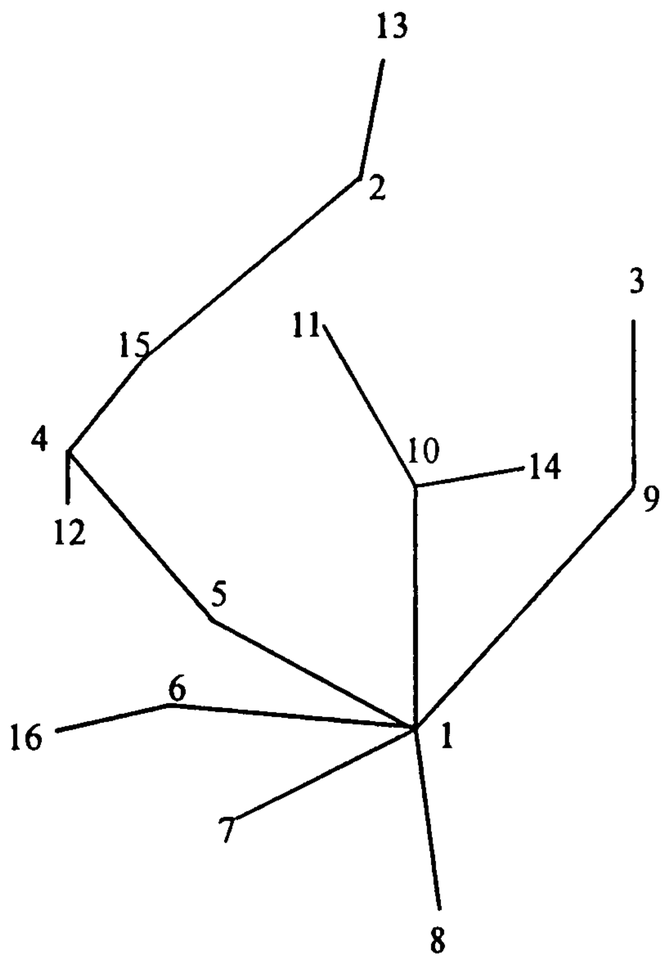
	Best known solution $c(T_{best,f})$	Links used for the best known solutions
Rothlauf's test problem 1	60,883	3-1, 4-1, 4-2, 5-1, 6-1, 7-1, 8-1, 9-3, 11-4, 11-10, 12-4, 13-2, 14-11, 15-4, 16-6
Rothlauf's test problem 2	58,619	5-1, 6-1, 7-1, 8-3, 10-1, 10-2, 10-3, 10-9, 11-4, 12-2, 13-10, 14-4, 14-10, 15-5
Rothlauf's test problem 3	28,451	5-1, 5-4, 6-1, 7-1, 8-1, 9-1, 9-3, 10-1, 11-10, 12-4, 13-2, 14-10, 15-2, 15-4, 16-6
Rothlauf's test problem 4	112,938	2-1, 3-1, 7-1, 7-5, 7-6, 8-1, 10-1, 11-1, 12-1, 12-4, 13-10, 14-9, 14-10, 15-1, 16-7



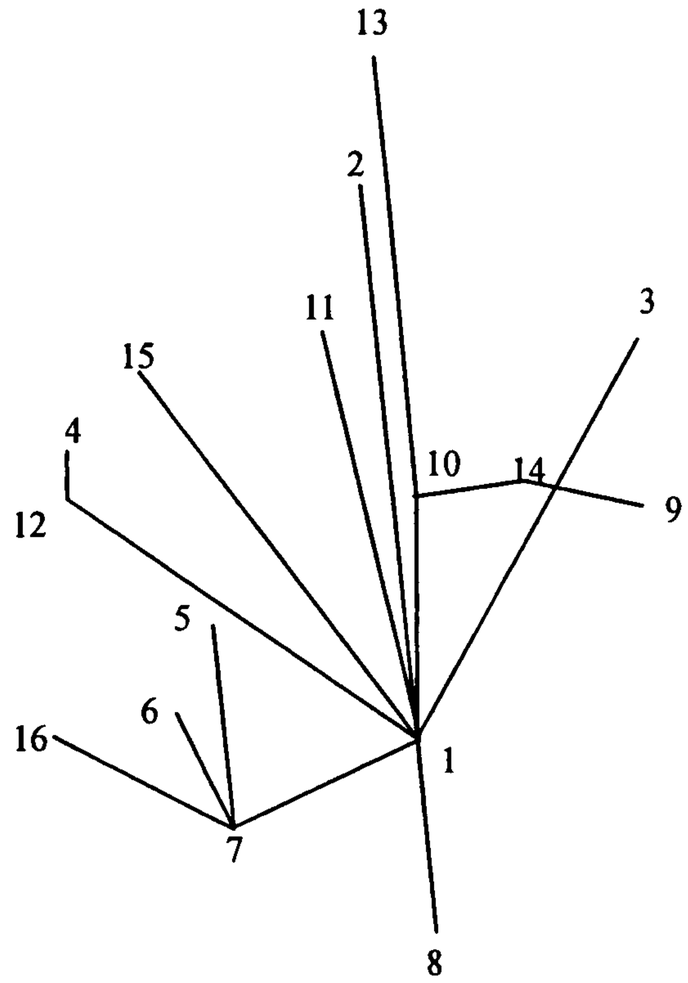
(a) Rothlauf 1



(b) Rothlauf 2



(c) Rothlauf 3



(d) Rothlauf 4

Figure 7.2 The best known network configurations of the four Rothlauf's test problems

7.6 Experimental Results

To investigate the performance of a GA with the Looped Local Search technique developed, the seven OCST benchmark problems (as detailed in the last section) have been used in this study. Experiments over 20 runs were carried out on each type of the benchmark problems. In the previous study on these seven OCST benchmark problems by various researchers, a wide range of population size from 16 to 2,000 had been used (Rothlauf, 2006; Thompson *et al.*, 2007). In order to keep it simple and test the performance of the developed optimisation algorithm with Looped Local Search technique, a population size of 20 (as used in Chapter 6) was adopted for all the seven benchmark problems in the present study. By inspection of the experimental results, the number of generations used for the test problems of Palmer, Raidl and Rothlauf 1 is 200; while 500 was used for test problems of Rothlauf 2, Rothlauf 3 and Rothlauf 4 to ensure convergence was reached. Due to the larger node numbers involved in Berry's test problem, the number of generations used was 1,000 for this problem. As the crossover and mutation rates used by the previous researchers had not been specified, crossover rate of 0.4 and mutation rate of 0.1 (as used in Chapter 6 for the DCS piping optimisation problem) were adopted here. It is because both the problem size (except the Berry's test problem) and the features of the OCST problem are similar to that of the DCS piping network in the present study. The results are shown in Tables 7.19 to 7.21. Table 7.19 lists the experimental results over 20 runs, including minimum values, average values, standard deviations (Std Dev) and success rates (Succ R) of Palmer's, Raidl's and Berry's test problems while that of the four Rothaluf's test problems are listed in Tables 7.20 and 7.21.

Table 7.19
Experimental results over 20 runs for Palmer's, Raidl's and Berry's test problems

	Palmer		Raidl		Berry	
Run no.	Best Function Value ($x10^3$)	First Hit Generation No.	Best Function Value	First Hit Generation No.	Best Function Value	First Hit Generation No.
1	3,429	193	157,570	187	17,375	364
2	3,627	196	157,570	181	16,915	348
3	3,429	189	157,570	194	16,915	464
4	3,429	194	157,570	183	16,915	502
5	3,429	159	164,544	157	18,687	389
6	3,492	199	157,570	187	16,915	481
7	3,429	169	157,570	181	16,915	395
8	3,429	140	165,248	194	16,915	569
9	3,429	150	157,570	183	17,375	438
10	3,429	159	157,570	157	16,915	625
11	3,820	142	157,570	138	16,915	498
12	3,429	157	178,284	185	17,375	611
13	3,429	182	157,570	167	16,915	359
14	3,429	145	157,570	138	16,915	528
15	3,429	153	157,650	157	19,039	566
16	3,429	176	157,570	168	16,915	682
17	3,429	149	161,604	143	16,915	559
18	3,720	173	157,570	177	16,915	475
19	3,429	169	157,570	162	16,915	382
20	3,429	145	157,570	154	17,785	519
Min.	3,429	140	157,570	138	16,915	348
Average	3,476	167	159,544	170	17,222	488
Std Dev	111.7	19.8	5,000.0	18.1	613.6	96.4
Succ R	80%	-	75%	-	70%	-

Table 7.20
Experimental results over 20 runs for Rothlauf's test problems (1) and (2)

Run no.	Rothlauf's Problem (1)		Rothlauf's Problem (2)	
	Best Function Value	First Hit Generation No.	Best Function Value	First Hit Generation No.
1	60,883	196	56,534	221
2	61,353	199	56,534	146
3	61,746	191	56,534	497
4	60,883	196	60,907	406
5	60,883	199	56,534	255
6	60,883	191	60,600	300
7	60,883	196	56,534	141
8	60,945	199	56,534	243
9	61,353	191	56,534	162
10	61,353	191	60,600	176
11	60,883	186	56,534	364
12	60,981	192	59,451	298
13	60,883	194	56,534	451
14	60,883	176	56,534	481
15	61,725	197	56,534	295
16	61,415	196	61,429	349
17	60,883	189	56,534	346
18	61,213	182	56,534	296
19	60,883	197	56,891	411
20	60,883	190	56,534	326
Min.	60,883	176	56,534	141
Average	61,090	192	57,568	308
Std Dev	296.1	5.9	1,826.9	107.8
Succ R	55%	-	70%	-

Table 7.21
Experimental results over 20 runs for Rothlauf's test problems (3) and (4)

Run no.	Rothlauf's Problem (3)		Rothlauf's Problem (4)	
	Best Function Value	First Hit Generation No.	Best Function Value	First Hit Generation No.
1	28,679	254	114,298	247
2	28,451	496	112,938	438
3	28,451	387	112,938	223
4	28,451	104	114,298	166
5	28,821	357	112,938	347
6	28,821	495	119,294	330
7	28,451	298	112,938	225
8	29,227	279	112,938	413
9	28,451	197	114,298	487
10	28,451	112	112,938	353
11	28,451	377	114,039	366
12	29,484	469	112,938	295
13	28,451	298	112,938	411
14	28,451	387	115,023	382
15	28,451	316	121,547	297
16	29,611	405	112,938	268
17	28,451	323	113,104	365
18	28,451	241	112,938	319
19	28,716	428	112,938	269
20	28,451	291	112,938	327
Min.	28,451	104	112,938	166
Average	28,662	326	114,058	326
Std Dev	364.8	111.2	2,300.7	80.1
Succ R	65%	-	60%	-

Experimental results of these seven benchmark problems conducted previously by various researchers using different encoding methods are listed for comparison with the results obtained in the present study. For all the benchmark problems, tournament selection without replacement of size 2 and uniform crossover were used. The results for test problems Palmer (12 nodes) and Berry (35 nodes) are listed in Tables 7.22 and

7.23 respectively (Rothlauf, 2006). For the four real-world tele-communication test problems (Rothlauf 1 to 4), the results are shown in Tables 7.24 to 7.27 (Rothlauf, 2006). The population size and maximum number of generation of all the test problems used by various researchers are listed inside these tables. There is no result of the Raidl's test problem provided by Rothlauf (2006) without reason stated. Therefore, published results from other researchers (Thompson *et al.*, 2007) including the Raidl's test problem are quoted in Table 7.28.

For comparison with the published results of the benchmark problems, a number of indices are used including P_{succ} *(%)*: the percentage of experimental runs which can search the best known solution; μ *(*σ*) of* $c(T_{best,f})$: the average value (standard deviation) of the best solutions obtained; and μ *(*σ*) of* t_{conv}: the average value (standard deviation) of the generation number at which the best solution is obtained.

Table 7.22
Performance of GA for Palmer's test problem (12 nodes) (Rothlauf, 2006)

	Encoding method	P_{succ} (%)	$c(T_{best,f})$ μ (σ)	t_{conv} μ (σ)
Palmer's problem (12 nodes) $c(T_{best,f})$ = 3,428,509 Population size: 300 Max. gen. no.: 200	Prüfer Number	0	4,111,091 (100,102)	46.1 (6.9)
	CV	0	3,700,839 (115,314)	200 (0)
	NB ($P_2 = 1$)	100	3,428,509 (0)	35.5 (5.2)
	NB ($P_2 = 20$)	60.4	3,463,583 (56,789)	33.6 (6.7)
	LB ($P_1 = 1$)	62.8	3,434,632 (14,913)	52.6 (7.0)
	LB ($P_1 = 20$)	35.2	3,450,763 (27,265)	60.2 (7.9)
	LBN ($P_2 = P_2 = 1$)	43.2	3,448,563 (22,482)	77.3 (11.1)
	NetKey	36.0	3,452,988 (27,768)	60.4 (7.2)
	NetDir	26.8	3,456,236 (38,339)	61.5 (8.8)
	Present Study	80	3,475,743 (111,716)	167 (19.8)

Table 7.23
Performance of GA for Berry's test problem (35 nodes) (Rothlauf, 2006)

	Encoding method	P_{succ} (%)	$c(T_{best,f})$ μ (σ)	t_{conv} μ (σ)
Berry's problem (35 nodes) $c(T_{best,f})$ = 16,915 Population size: 300 Max. gen. no.: 200	Prüfer Number	0	59,553 (2,795)	104.7 (12.3)
	CV	0	120,397 (12,749)	200 (0)
	NB ($P_2 = 1$)	98.8	16,916 (12)	56.1 (6.4)
	NB ($P_2 = 20$)	0	20,769 (1,976)	45.6 (5.9)
	LB ($P_1 = 1$)	100	16,915 (0)	75.2 (6.8)
	LB ($P_1 = 20$)	11.2	17,519 (526)	94.0 (8.0)
	LBN ($P_2 = P_2 = 1$)	25.6	17,181 (250)	128.2 (12.7)
	NetKey	2.8	18,083 (763)	99.1 (8.6)
	NetDir	0	27,975 (4,019)	200 (0)
	Present Study	70	17,222 (614)	488 (96.4)

Table 7.24
Performance of GA for Rothlauf's test problem (1) (16 nodes) (Rothlauf, 2006)

	Encoding method	P_{succ} (%)	$c(T_{best,f})$ μ (σ)	t_{conv} μ (σ)
Rothlauf's problem(1) (16 nodes) $c(T_{best,f})$ = 60,883 Population size: 800 Max. gen. no.: 200	Prüfer Number	0	63,680 (236)	62.1 (14.1)
	CV	0	71,657 (2,592)	200 (0)
	NB (P_2 = 1)	0	64,654(684)	34.2 (9.6)
	NB (P_2 = 20)	0	64,997 (106)	28.6 (5.8)
	LB (P_1 = 1)	31.2	61,202 (424)	85.1 (8.5)
	LB (P_1 = 20)	27.2	61,161 (372)	85.5 (8.4)
	LBN (P_2 = P_2 = 1)	0.4	61,233 (123)	119.3 (17.5)
	NetKey	29.6	61,144 (345)	86.6 (9.1)
	NetDir	0	61,709 (523)	110.9 (11.1)
	Present Study	55	61,090 (296)	192 (5.9)

Table 7.25
Performance of GA for Rothlauf's test problem (2) (15 nodes) (Rothlauf, 2006)

	Encoding method	P_{succ} (%)	$c(T_{best,f})$ μ (σ)	t_{conv} μ (σ)
Rothlauf's problem(2) (15 nodes) $c(T_{best,f})$ = 58,619 Population size: 2,000 Max. gen. no.: 200	Prüfer Number	0	66,221 (868)	63.9 (28.2)
	CV	0	69,598 (1,960)	200 (0)
	NB (P_2 = 1)	0	69,901 (2,421)	45.7 (14.2)
	NB (P_2 = 20)	0	73,276 (917)	34.4 (8.1)
	LB (P_1 = 1)	72	58,664 (114)	90.9 (9.6)
	LB (P_1 = 20)	76	58,652 (61)	90.2 (8.5)
	LBN (P_2 = P_2 = 1)	0	59,078 (318)	132.3 (18.1)
	NetKey	75.6	58,651 (57)	91.1 (9.0)
	NetDir	16.8	58,872 (231)	107.9 (7.2)
	Present Study	70	57,568 (1,827)	308 (107.8)

Table 7.26
Performance of GA for Rothlauf's test problem (3) (16 nodes) (Rothlauf, 2006)

	Encoding method	P_{succ} (%)	$c(T_{best,f})$ μ (σ)	t_{conv} μ (σ)
Rothlauf's problem(3)	Prüfer Number	0	31,274 (500)	79.0 (23.0)
(16 nodes)	CV	0	34,870 (936)	200 (0)
$c(T_{best,f})$ = 28,451	NB (P_2 = 1)	0	32,704 (897)	45.7 (16.2)
Population size: 1,200	NB (P_2 = 20)	0	34,286 (573)	34.3 (8.1)
Max. gen. no.: 200	LB (P_1 = 1)	41.6	28,709 (337)	97.6 (9.8)
	LB (P_1 = 20)	48.0	28,694 (351)	99.4 (10.7)
	LBN (P_2 = P_2 = 1)	0	30,386 (298)	157.2 (21.3)
	NetKey	44	28,722 (352)	98.2 (9.1)
	NetDir	38.4	28,891 (395)	115.2 (13.3)
	Present Study	65	28,662 (364)	326 (111.2)

Table 7.27
Performance of GA for Rothlauf's test problem (4) (16 nodes) (Rothlauf, 2006)

	Encoding method	P_{succ} (%)	$c(T_{best,f})$ μ (σ)	t_{conv} μ (σ)
Rothlauf's problem(4)	Prüfer Number	0	128,572 (2,350)	68.2 (29.0)
(16 nodes)	CV	0	146,387 (4,592)	200 (0)
$c(T_{best,f})$ = 112,938	NB (P_2 = 1)	0	125,095 (1,856)	52.4 (45.7)
Population size: 800	NB (P_2 = 20)	0	128,211 (1,260)	28.6 (17.2)
Max. gen. no.: 200	LB (P_1 = 1)	13.6	114,741 (2,570)	94 (11.6)
	LB (P_1 = 20)	14	115,106 (2,790)	93.6 (11.0)
	LBN (P_2 = P_2 = 1)	0	117,755 (1,990)	132.1 (21.9)
	NetKey	12.8	115,016 (2,968)	94 (8.2)
	NetDir	15.6	113,086 (438)	92.2 (12.3)
	Present Study	60	114,058 (2,300)	326 (80.1)

Table 7.28

Performance of GA for Palmer 12, Raidl 20 and Berry 35 problems (Thompson *et al.*, 2007)

OCST		Size *n* of Problem Instance		
		12 (Palmer)	20 (Raidl)	35 (Berry)
Prüfer Number	minimum cost	3,428,509	157,570	19,451
	opt. found	54.6%	0.1%	0.0%
	average cost	3,445,837	216,885	23,289
	average gen.	50.67	134.96	256.59
NetKeys	minimum cost	3,428,509	157,570	16,915
	opt. found	36.4%	63.9%	92.6%
	average cost	3,451,827	159,278	16,930
	average gen.	61.75	128.17	209.93
Direct Tree code	minimum cost	3,428,509	157,570	16,915
	opt. found	60.9%	81.1%	99.9%
	average cost	3,438,254	158,647	16,915
	average gen.	46.67	95.91	153.98
Present Study	minimum cost	3,428,509	157,570	16,915
	opt. found	80.0%	75.0%	70.0%
	average cost	3,475,743	159,544	17,222
	average gen.	167	170	488

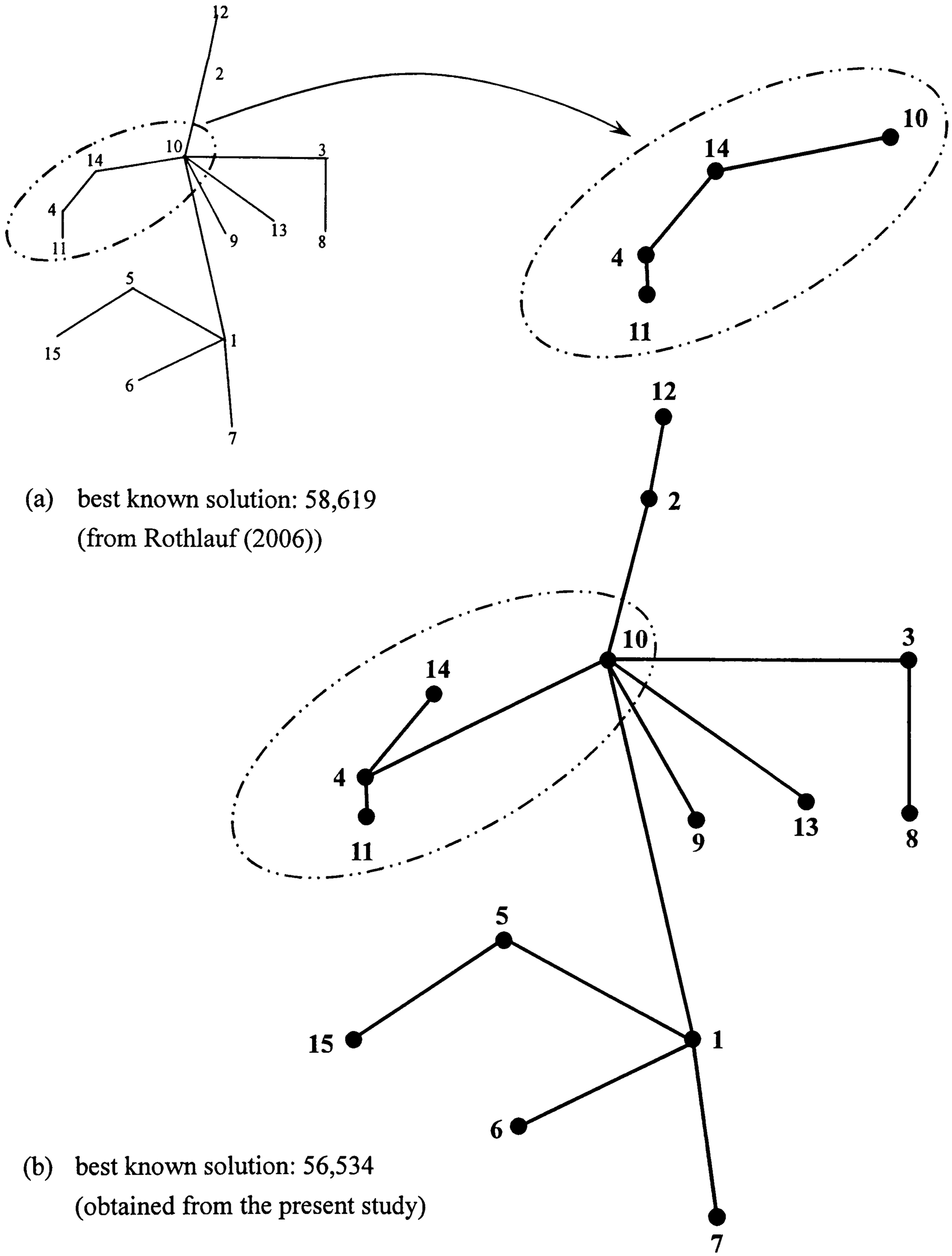
In Table 7.22, the results of the simplest test problem (Palmer, 12 nodes) are shown. NB encoding method with $P_2 = 1$ showed the best performance of $P_{succ} = 100\%$ while Prüfer Number and CV failed to find the best known solution in all the experimental runs. The result from the present study gave a value of $P_{succ} = 80\%$ and $c(T_{best,f})_{\mu} = 3,475,743$ which are competitive with the performance of the NB encoding since the population size in the present study (20) is much smaller than that used by Rothlauf (300).

The results of the test problem with the largest node number (Berry) are shown in Table 7.23. Again, NB and LB encoding methods with small values of P_2 and P_1 can find the

best known solution at high success rate while Prüfer Number and CV, NB with high value of P_2 and NetDir have poor performance of zero success rate. The results from the present study were fairly good with P_{succ} = 70% and $c(T_{best,f})_{\mu}$= 17,222 without the need to assign any guessed values of P_1 or P_2, which is solution dependent.

The results from the present study for test problems of Rothlauf 1, Rothlauf 3 and Rothlauf 4 outperformed the published results from Rothlauf (2006) as shown in Table 7.24, 7.26 and 7.27.

For the test problem Rothlauf 2, besides a high success rate of 70% obtained from the present study, there is an encouraging discovery that a "better" best known solution of 56,534 (see Table 7.20) was found from the present study which is better than the best solution of 58,619 (see Table 7.25) found by Rothlauf (2006). This result can be further illustrated by Figures 7.3 (a) and (b). Figure 7.3 (a) shows the configuration of the communication network with the "past" best known solution (58,619). Through the performance of Looped Local Search developed in this study, the original link 10 – 14 in Figure 7.3 (a) was selected and node 10 was used as a searching point and then was connected to node 4 (see Figure 7.3 (b)) so that an improved best known solution of 56,534 was obtained.



(a) best known solution: 58,619
(from Rothlauf (2006))

(b) best known solution: 56,534
(obtained from the present study)

Figure 7.3 Configurations of the best known communication networks (Rothlauf 2)

(a) from Rothlauf (2006) and (b) from the present study

The performance of the GA for the test problem of Palmer (12 nodes), Raidl (20 nodes) and Berry (35 nodes) conducted by Thompson *et al.*, (2007), with different encoding methods, is shown in Table 7.28. It can be seen that the Prüfer Number encoding method can search the best known solution for the simpler test problems (Palmer, 12 nodes; Raidl, 20 nodes). The NetKeys and Direct Tree code methods can perform well for test problems with larger node numbers (Berry, 35 nodes). The results from the present study can search the best known solutions for all the three types of test problems with fairly good percentage of success.

7.7 Concluding Remarks

In this chapter, the experimental results of the seven benchmark test problems (using a GA with the Looped Local Search technique) were compared with the published results from various researchers. It was found that the algorithm developed in the present study can outperform the methods reported in previous research with different encoding methods in which some of them need input of solution-dependent parameters. With the simple Integer String encoding method and the Looped Local Search technique, the best known solution can be found efficiently at comparatively smaller population size and shorter computational time. Moreover, in one of the benchmark problems (Rothlauf 2), a "better" best known solution was found by the present study. The results are encouraging and suggest that the Looped Local Search method developed in this work is an effective and efficient tool for optimal network design of the distribution piping system in DCS as well as in the communication spanning tree problem.

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

In the present study, a detailed investigation of the optimisation of distribution piping network in a district cooling system using a genetic algorithm has been carried out. Four major tasks were carried out: (i) developing typical climatic data for building thermal energy simulations; (ii) developing a local search method for optimisation using a genetic algorithm; (iii) identifying the crucial factors in optimal design of distribution piping configuration in district cooling systems; (iv) validation of the performance of the algorithm developed with benchmark optimal communication spanning tree problems. This chapter summarises the major findings of the research in this study, explains the limitations of the work; and recommends areas for further study.

8.1 Summary of Major Findings

The results from this research can make a contribution to more effective and efficient design of optimal piping configuration in district cooling system (DCS). The major research findings are summarised in the following paragraphs.

(i) Typical Meteorological Year

In order to evaluate the pumping energy of the distribution piping network in a DCS, it is necessary to determine the hourly chilled water demand of each building category in a city development by running building thermal energy simulations. Weather data is one of the key factors for successful building energy simulation, and building simulation software requires hourly weather data such as dry bulb

temperature, dew point temperature, solar radiation, wind speed and direction. Since weather conditions can vary significantly from year to year, there is a need to derive a dataset of typical weather year (TWY) to represent the long-term typical weather condition over a year. Through a literature review, it is found that the typical meteorological year (TMY) is the most commonly used and appropriate one for parametric study in building thermal energy analysis. In the present work, Hall *et al.*'s procedures (1978) and IWEC (International Weather Year for Energy Calculation) weighting factors, which are in line with the international usage and which give equal importance to dry bulb temperature and solar radiation, were used to develop a TMY for Hong Kong.

The method used to select a TMY for a given location involves selecting, by statistical methods, one typical meteorological month (TMM) for each of the 12 calendar months from a period of years and concatenating the 12 months to form a TMY. This is done by comparing the cumulative distribution function (CDF) for each year with the CDF for the long-term composite of all the years in a period for four major weather indices including dry bulb temperature, dew point temperature, wind speed and solar radiation. The statistics used to measure the closeness of each year's CDF to the long-term composite for a given index is the Finkelstein-Schafer (*FS*) statistics (Finkelstein & Schafer, 1971). This statistic is the average absolute difference between the yearly CDF and the long-term composite CDF. For each year, there are statistics computed for each index. A weighted sum, using various weighting factors, is calculated for the indices and this sum is used to select potential candidates for the TMM. Twelve TMMs from different years form a TMY.

In this study, hourly measured weather data of years 1979-2003 were acquired from the Hong Kong Observatory. The weather data include dry bulb temperature, dew point temperature, wind speed, global horizontal solar radiation, relative humidity, wind direction, sky cover, atmospheric pressure and rainfall. The first four sets of weather data were used for selecting TMMs and, together with the remaining sets of data, to construct a TMY for Hong Kong in the IWEC format. The 12 TMMs developed are evenly spread over the 25-year period. Five TMMs are found from 80s, another four from 90s and the remaining three are found from 2000, 2002 and 2003, respectively.

The TMY developed in this study has been accepted by the U. S. Department of Energy (DOE) as one of the mostly requested weather data and is now available in EnergyPlus weather format from the official web site of the Building Technologies Program, Department of Energy (DoE, 2007).

For assessing the typicality of the TMY generated, computer simulations using weather data from the generated TMY and the individual years from 1979 to 2003 were performed as a comparative study. From the analysis of the predicted electricity consumption for different individual years, it is shown that the TMY developed in this study can give a good indication of long-term energy performance in building energy simulation. The TMY weather dataset developed in this study has applications beyond the optimisation of piping configuration in DCS, but it is also useful for building energy simulation by other researchers as there is no other systematically generated and representative weather data available locally.

(ii) Local Search Method for Optimisation

The GA performs an adaptive and global sampling of the search domain. This type of heuristic algorithm for solving optimisation problems can find near-optimal solutions within a reasonable amount of computation time. Researchers mainly make use of the genetic operators, crossover and mutation, in a GA, but better solutions in the vicinity of the current near-optimal solution may be missed if the search process just relies on probabilistic search heuristics. In order to avoid missing possible better solutions, a local search algorithm was developed and incorporated into the GA for finding better solutions, generally starting with a feasible solution found by the crossover/mutation operators. At each iteration, an improved solution could be found by searching the neighborhood of the current solution.

In the present study, a Local Search/Looped Local Search technique was developed for the optimisation of distribution piping configuration in a DCS. During the GA optimisation process, if the function value converged and there was no further improvement by the GA operators (crossover & mutation) after x generations (arbitrarily assigned), a subroutine "Local Search/Looped Local Search" was called. The elite in the current generation was taken as the candidate for the Local Search/Looped Local Search. In each step, the longest link of the elite and the two nodes joining this link were identified. One of the two identified nodes was taken as a searching point and connected to one of the remaining nodes to form a new link. This new link replaced the original longest link to form a new piping network. If the function value of this new network was smaller than that of the original elite, the newly formed circuit replaced the original 'best known' circuit.

The Local Search repeats by connecting the searching node to other remaining nodes in the circuit. If there was no further improvement by the Local Search, Looped Local Search was called, in which the second longest link of the elite and the two nodes joining the link were identified. Similar search steps as those in Local Search were performed. The procedures repeat for the other links, in descending order of the link-length, until the last link has been selected to undergo the Local Search process. By computational experiments, it was demonstrated that there were promising improvements in GA performance, in terms of both solution quality and computational efficiency.

(iii) Critical Factors in Optimisation of Piping Configuration

In the optimisation of distribution piping configuration by GA, the effect on the search performance by a number of factors have been investigated. The factors include:

- the effectiveness of Local Search and Looped Local Search implemented in the GA optimisation process;
- the application of a penalty value versus repair algorithm for infeasible candidates;
- using elite or randomly selected population as the candidate to undergo a Local Search/Looped Local Search;
- link selection criteria including "longest length", "minimum diameter" and "critical path";
- the appropriate frequency of local search (FoLS);
- the appropriate stage to start a local search.

Through a series of parametric experiments, it was found that a Looped Local Search using elite candidate with repair algorithm can improve both the solution quality and computational efficiency. Amongst the three different link-selection criteria used in Looped Local Search, namely "longest length", "minimum diameter" and "critical path", it was found that the first was the best choice for implementing the Looped Local Search. For frequency of local search (FoLS), a value of ten (> 10) was found to be a good basis for calling the Looped Local Search subroutine. Moreover, there was no evidence that starting the Looped Local Search after a certain number of generations can result in a better performance of the GA. These findings are very useful and can facilitate an effective and efficient environment for conducting optimal design of the distribution piping configuration in a DCS.

(iv) Validation of the Effectiveness of the Developed Algorithm using Benchmark Problems

In order to validate the effectiveness of the developed Looped Local Search technique used with GA, benchmark test problems were selected and tested with the algorithm developed and compared with the published results from various researchers. The test problem chosen was the optimal communication spanning tree (OCST) problem, the objective of which is to find a network to connect all the given nodes (demand points) and satisfy their communication requirements at a minimum total cost. Seven different OCST benchmark problems were used, including Palmer's 12-node OCST problem, Raidl's 20-node OCST problem, the Berry 35-node OCST problem and Rothlauf's four different real-world OCST problems.

It was found that the method developed in this study can, on average, outperform the results of the previous work by other researchers with different encoding methods, which in some cases need the input of solution-dependent parameters. With the Looped Local Search technique developed, the best known solution was found efficiently using a smaller population size and producing shorter computational times. Moreover, in one of the benchmark problems (Rothlauf's 2nd real-world OCST problem), a "better" best known solution was found in this study. The results are encouraging and demonstrate that the Looped Local Search method developed is an effective and efficient tool for the optimal design of the distribution piping system in a DCS, as well as for the general communication spanning tree problem.

8.2 Limitations

There are some deficiencies in this study which are listed below.

(i) Weighting Factors Used for Deriving TMY

In this study, IWEC weighting factors, which are in line with international usage and which give equal importance to dry bulb temperature and solar radiation, were used to develop a TMY for Hong Kong. These weighting factors may be location/building-type dependent since the relative importance of various weather indices on the energy performance of buildings are influenced by different characteristics under different building designs and different climatic conditions.

(ii) Encoding Method for Representing Piping Configurations

The basic requirements of an appropriate encoding method used in the present

study are simplicity, ability to represent all the possible piping configurations and non-bias to any type of piping configuration. Both Prüfer Number and Integer String encoding methods can fulfill these requirements. Due to the feature of low locality of Prüfer Number method, this type of encoding was not adopted. The Integer String encoding method has been selected for representing the piping configurations in this study. However, since the Integer String method cannot avoid representing infeasible piping configurations, extra computational time was spent in repairing infeasible candidates during the optimisation process.

(iii) Constraints on a Piping Network

The locations of the consumer buildings (nodes) in a hypothetical site were pre-fixed in this study. It was assumed that there was no constraint on the site so that the piping segments can be connected directly between nodes without any obstructions. This may not be the case in real applications in which longer pipework may be needed for connecting nodes in order to avoid obstructions and/or tee joints may be installed at pre-determined locations. During GA optimisation, these additional constraints will have to be taken into consideration.

8.3 Recommendations for Future Work

Based on the findings from this research study and the limitations discussed above, future research and development work are recommended as follows.

(i) Fine Tune the TMY Weather Dataset

The weighting factors of various weather indices used in generating the TMY

weather dataset can be further investigated so that the relative importance of various weather indices can be truly reflected for different applications such as building types, system designs, local climatic conditions, etc. Moreover, more hourly weather data of different years, if available, should be used to generate the TMY dataset so that the weather file can be more representative.

(ii) Improve the Features of Encoding Method

Both Prüfer Number and Integer String encoding methods are simple and capable to encode all the possible piping configurations without bias. The Prüfer Number method has limitations due to its low locality, while the Integer String method needs extra computational time for repairing infeasible piping configurations. There is a potential need to develop a new encoding method which combines the advantages of these two existing encoding methods. It is expected that, with the new encoding method with improved features, the effectiveness and efficiency of the developed Local Search/Looped Local Search in the present study can be further improved.

(iii) Incorporate Spatial Constraints into the Optimisation Problem

It is beneficial to have a real site which contains groups of scattered DCS consumer buildings with constraints such as obstructions, tee joints at pre-set locations, etc. The investigation could be extended to develop an effective algorithm to solve this problem.

REFERENCES

Ahn, T., 1993. *Optimal Design of Municipal and Irrigation Water Distribution Systems.* PhD Thesis, Virginia Polytechnic Institute and State University.

Ahuja, R.K., Magnanti, T.L., Orlin, J.B., 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall.

Altinbilek, H., 1981. *Optimum design of branched water distribution networks by linear programming*, Proceedings of the International Symposium on Urban Hydrology, Hydraulics and Sediment Control, Lexington, KY, 249-254.

ASHRAE, 2002. *International Weather for Energy Calculations (IWEC Weather Files) User's Manual*, Version 1.1.

Ayyub, B.M., McCuen, R.H., 1996. *Numerical Methods for Engineers.* Prentice Hall.

Babayan, A., Zoran, K., Savic, D., Walters, G., 2006. *Comparison of two methods for the stochastic least cost design of water distribution systems*. Engineering Optimization, 38(3): 281-297.

Babus'Haq, R.F., Probert, S.D., 1987. *Optimal configuration for horizontal double-pipe district heating/cooling distribution networks*, ASHRAE Technical Data Bulletin, District Heating and Cooling, 160-175.

Babus'Haq, R.F., Probert, S.D., George, H.E., 1990. *District heating and/or district cooling distribution pipelines: optimal configurations*, Proc Instn Mech Engrs, 204:57-66.

Bäck, T., 1996. E*volutionary Algorithms in Theory and Practice.* Oxford University Press.

Bäck, T., Fogel, D.B., Michalewicz, T., 2000. *Evolutionary Computation 1*. Institute of Physics Publishing.

Bahadoorsingh, S., Milanovic, J.V., Zhang, Y., Cupta, C.P., Dragovic, J., 2007. *Minimization of Voltage Sag Costs by Optimal Reconfiguration of Distribution Network Using Genetic Algorithms*. IEEE Transactions on Power Delivery, 22(4): 2271-2278.

Bahnfleth, W.P., Joyce, W.S., 1994. *Energy use in a district cooling system with stratified chilled-water storage*, ASHRAE Transactions, 100(1): 1767-1778.

Baker, J.E., 1987. *Reducing bias and inefficiency in the selection algorithm.* Proc. 2nd Int. Conf. on Generic Algorithms, 14-21.

Benonysson, A., Bøhm, B., Ravn, H.F., 1995. *Operational optimization in a district heating system*, Energy Conversion and Management, 36(5): 297-314.

Bernotat, K., Sandberg, T., 2004. *Biomass fired small-scale CHP in Sweden and the Baltic States: a case study on the potential of clustered dwellings*, Biomass & Bioenergy, 27:521-530.

Berry, L.T.M., Murtagh, B.A., McMahon, G., 1995. *Applications of a genetic-based algorithm for optimal design of tree-structured communication networks.* Proceedings of the Regional Teletraffic Engineering Conference of the International Teletraffic Congress, Pretoria, South Africa, 361-370.

Bloomster, C.H., Fassbender, L.L., 1983. *Simulation and Analysis of District Heating and Cooling Systems*. Proceedings of the International District Heating Association Conference, 19-22 June 1983, Pennsylvania, 1-25.

Bogdan, Ž., Kopjar, D., 2006. *Improvement of the cogeneration plant economy by using heat accumlator*, Energy, 31:1949-1956.

Brown, R.E., 2003. *Network Reconfiguration for Improving Reliability in Distribution Systems.* Proceedings of the Power Engineering Society General Meeting, IEEE, 13-17 July. 4:2419-2424.

Burer, M., Tanaka, K., Favrat, D., Yamada, K., 2003. *Multi-criteria optimization of a district cogeneration plant integrating a solid oxide fuel cell-gas turbine combined cycle, heat pumps and chillers*, Energy, 28:497-518.

Calhoun, C., 1971. *Optimization of pipe systems by linear programming*, Control of Flow in Closed Conduits, Colorado State University, Ft. Collins, 175-192.

Cayley, A., 1889. *A theorem on trees.* Quarterly Journal of Mathematics, 23:376-378.

Cela, E., 1998. *The Quadratic Assignment Problem, Theory and Algorithms*. Kluwer Academic Publishers.

Chan, A.L.S., Chow, T.T., Fong, S.K.F., Lin, J.Z., 2006a. *Generation of a typical meteorological year for Hong Kong*, Energy Conversion and Management. 47:87-96.

Chan, A.L.S., Chow, T.T., Fong, S.K.F., Lin, J.Z., 2006b. *Performance evaluation of district cooling plant with ice storage*, Energy, 31:2414-2426

Chan, A.L.S., Hanby, V.I., Chow, T.T., 2007a. *Optimization of distribution piping network in a district cooling system using genetic algorithm with local search,* Energy Conversion and Management, 48(10), pp. 2622-2629.

Chan, A.L.S., Hanby, V.I., Chow, T.T., 2007b, *Application of Genetic Algorithm with Local Search in Optimal Piping Network Design of a District Cooling System,* Proceedings of the 6th International Conference on Indoor Air Quality, Ventilation & Energy Conservation in Buildings, Sendai, Japan, 28-31 October, 2007.

Chen, Q., 2004. *Simulation of Thermal Plant Optimization and Hydraulic Aspects of Thermal Distribution Loops for Large Campuses.* PhD Thesis, Texas A & M University.

CLP, 2006. *Tariff Information.* China Light and Power Holdings Limited. (web site: http://www.clpgroup.com/)

Crawley, D.B., 1998. *Which Weather Data Should You Use for Energy Simulations of Commercial Buildings?* ASHRAE Transactions. 104:498-515.

Dandy, G., Simpson, A., Murphy, L., 1996. *An improved genetic algorithm for pipe network optimization*, Water Resources Research, 32(2): 449-458.

Dengiz, B., Altiparmak, F., Smith, A.E., 1997. *Local Search Genetic Algorithm for Optimal Design of Reliable Networks*. IEEE Transactions on Evolutionary Computation, 1(3): 179-188.

Deo, N., Abdalla, A., 2000. *Computing a Diameter-Constrained Minimum Spanning Tree in Parallel*. Proceedings of the 4th Italian Conference on Algorithms and Complexity, CIAC 2000, 17-31.

DoC, 1980. *DOE-2 Reference Manual Part 2*, US Department of Commerce, National Technical Information Services, Springfield, Chapter VIII.

DoE, 2005. *EnergyPlus Manual*. Version 1.2.2, Department of Energy, US.

DoE, 2007. *EnergyPlus Energy Simulation Software, Weather Data*, http://www.eere.energy.gov/buildings/energyplus/cfm/weather_data3.cfm/region=2_asia_wmo_region_2/country=CHN/cname=China

Dotzauer, E., 2003. *Experiences in mid-term planning of district heating systems*, Energy, 28:1545-1555.

Eiben, A.E., Smith, J.E., 2003. *Introduction to Evolutionary Computing*. Springer.

Fanger, P.O., 1972. *Thermal Comfort Analysis and Applications in Environmental Engineering*, McGraw-Hill.

Farmani, R., Wright, J.A., Savic, D.A., Walters, G.A., 2005. *Self-Adaptive Fitness Formulation for Evolutionary Constrained Optimization of Water Systems.* Journal of Computing in Civil Engineering, 19(2): 212-216.

Finkelstein, J.M., Schafer. R.E., 1971. *Improved goodness-of-fit tests.* Biometrika 58:3:641-645.

Fletcher, R., Reeves, R.M., 1964. *Function Minimization by Conjugate Gradients*, The Computer Journal, 7:149-180.

Fragiadakis, M., Lagaros, N.D., Papadrakakis, M., 2006. *Performance-based multiobjective optimum design of steel structures considering life-cycle cost.*

Struct Multidisc Optim, 32:1-11.

Fu, L., Jiang, Y., Yuan, W., Qin, X., 2001. *Influence of supply and return water temperatures on the energy consumption of a district-cooling system*, Applied Thermal Engineering, 21:511-521.

Garey, M.S., Johnson, D.S., 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, New York.

Gaube, T., Rothlauf, F., 2001. *The Link and Node Biased Encoding Revisited: Bias and Adjustment of Parameters*. Applications of Evolutionary Computing, Springer, 1-10.

Gen, M., Li, Y., Ida, K., 1999. *Solving Multi-Objective Transportation Problem by Spanning Tree-Based Genetic Algorithm*. IEICE Trans. Fundamentals, E82-A: 12:2802-2810.

Glover, F., Laguna, M., 1997. *Tabu Search*. Kluwer Academic Publishers.

Gottlieb, J., Eckert, C., 2000. *A comparison of Two Representations for the Fixed Charge Transportation Problem.* Proceedings of the 6th International Conference on Parallel Problem Solving from Nature, Paris, France, September 18-20, 346-354.

Groot, C., Würtz, D., Hoffmann, K.H., 1990. *Optimizing Complex Problems by Nature's Algorithms: Simulated Annealing and Evolution Strategy – a Comparative Study.* Parallel Problem Solving from Nature, Springer-Verlag: 445-454.

Hall, I.J., Prairie, R.R., Anderson, H.E., Boes, E.C., 1978. *Generation of a Typical Meteorological Year.* Proceedings of the 1978 Annual Meeting of the American Section of the International Solar Energy Society, 669-671.

Hasnain, S.M., 1998. *Review on sustainable thermal-energy storage technologies, Part II: cool thermal storage*, Energy Conversion and Management, 39(11):1139-1153.

Haupt, R.L., Haupt, S.E., 2004. *Practical Genetic Algorithms.* 2nd edition, John Wiley

& Sons.

Henning, D., Amiri, S., Holmgren, K., 2006. *Modelling and optimisation of electricity, steam and district heating production for a local Swedish utility,* European Journal of Operational Research, 175:1224-1247.

HKSAR, 1995. *Code of Practice for Overall Thermal Transfer Value in Buildings 1995.* Buildings Department, Hong Kong Special Administrative Region Government.

HKSAR, 2003. *Performance-based Energy Code*. Electrical and Mechanical Services Department, Hong Kong Special Administrative Region Government.

HKSAR, 2006. *Economic background.* Economic Analysis Division, Financial Services Bureau, Hong Kong Special Administrative Region Government.

Hopcroft, J.E., Ullman, J.D., 1973. *Set merging algorithms.* SIAM J. Comput. 2:296-303.

Huang, J., 1998. *The Impact of Different Weather Data on Simulated Residential Heating and Cooling Load.* ASHRAE Transactions. 104:516-527.

Iancu, P., Plesu, V., Lavric, V., 2006. *Cost versus network length criteria in water network optimal design.* Computer Aided Chemical Engineering, 21(2): 1821-1826.

Ishibuchi, H., Murata, T., Tomioka, S., 1997. *Effectiveness of Genetic Local Search Algorithm.* Proceedings of the Seventh International Conference on Genetic Algorithms, July 19-23. 505-512.

Ishii, H., Shiode, S., Nishida, T., Namasuya, Y., 1981. *Stochastic spanning tree problem.* Discrete Applied Mathematics, 3:263-273.

Ito, T., 1999. *A genetic algorithm approach to piping route path planning*. Journal of Intelligent Manufacturing, 10:103-114.

Itoh, Y., Nagata, H., Liu, C., Nishikawa, K., 2000. *Comparative Study of Optimized and Conventional Bridges: Life Cycle Cost and Environmental Impact*. Proceedings

of US-Japan Workshop on Life-Cycle Cost Analysis and Design of Civil Infrastructure Systems, Hononlulu, Hawaii, 130-148.

Kasten, F., 1981. *Measurement and analysis of solar radiation data.* Energy and Buildings, 3(1):1-29.

Keeble, E., 1990. *Availability of UK Climatic Data for Use in Simulation*, BEPAC Technical Note 90/1, Building Research Establishment, October.

Kessels, J.F.A., Jonker, A.S., Akkerman, R., 2007. *Optimising the flow pipe arrangement for resin infusion under flexible tooling.* Composites, Part A: applied science and manufacturing, 38:2076-2085.

Khan, K.H., Rasul, M.G., Khan, M.M.K., 2004. *Energy conservation in buildings: cogeneration and cogeneration coupled with thermal-energy storage*, Applied Energy, 77:15-34.

Kitanovski, A., Poredos, A., 2002. *Concentration distribution and viscosity of ice-slurry in heterogeneous flow*, International Journal of Refrigeration, 25:827-835.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. *Optimization by Simulated Annealing,* Science, 220(4598): 671-680.

Knowles, J.D., Corne, D.W., 2001. *A Comparison of Encodings and Algorithms for Multiobjective Minimum Spanning Tree Problems.* Proceedings of the 2001 Congress on Evolutionary Computation, 1:544-551.

Kreher, D.L., Stinson, D.R., 1998. *Combinatorial Algorithms, Generation, Enumeration and Search.* CRC Press.

Lam, C., 2004. *The Development of District Cooling Systems in Hong Kong,* Proceedings of the 2004 Shenyang-Hong Kong Joint Symposium on Healthy Building in Urban Environment, 29-30 July 2004, Shenyang, China: A48-A56.

Lansey, K., Mays, L., 1989. *Optimal design of water distribution system design*, ASCE Journal of Hydraulic Engineering, 115(10): 1401-1418.

Lavic, V., Iancu, P., Plesu, V., 2007. *Cost-based desing of wastewater network optimal*

topology. Resources Conservation & Recycling. 50:186-201.

Li, D.H.W., Lam, J.C., 2001. *Analysis of solar heat gain factors using sky clearness index and energy implications.* Energy Conversion & Management. 42:555-571.

Loganathan, G., Greene, J., Ahn, T., 1995. *Design heuristic for globally minimum cost water distribution systems*, ASCE Journal of Water Resources Planning and Management, 121(2): 182-192.

Mariano-Romero, C.E., Alcocer-Yamanaka, V.H., 2005. *Multiogjective Optimization of Water-Using Systems*. Real-World Multi-Objective System Engineering, Nova Science Publishers, 163-192.

Marion W., Urban K., 1995. *User's Manual for TMY2s Derived from the 1961-1990 National Solar Radiation Data Base,* National Renewable Energy Laboratory.

Maxwell, E.L., 1998. *METSTAT the solar radiation model used in the production of the national solar radiation data (NSRDB)*. Solar Energy 62(4), 263-279.

Meesters K.P.H., Van Groenestijn, J.W., Gerritse, J., 2003. *Biofouling reduction in recirculating cooling systems through biofiltration of process water*, Water Research, 37:525-532.

Merz, P., Freisleben, B., 1997. *A Genetic Local Search Approach to the Quadratic Assignment Problem*, Proceedings of the Seventh International Conference on Genetic Algorithms, Michigan State University, East Lansing, MI, July 19-23, 465-472.

Metropoli, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E., 1953. *Equation of state calculations by fast computing machines*. J. Chem. Phys., 21:1087-1092.

Michalewicz, Z., 1996. *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd edition, Springer.

Michalewicz, Z., Fogel, D.B., 2004. *How to Solve It: Modern Heuristics*, 2nd edition, Springer.

Morley, M.S., Atkinson, R.M., Savic, D.A., Walters, G.A., 2001. *GAnet: genetic algorithm platform for pipe network optimisation.* Advances in Engineering Software, 32:467-475.

Nedjah, N., Mourelle, L.M., 2005. *Real-World Multi-Objective System Engineering*, Nova Science Publishers, New York.

Obara, S., 2006. T*he hot-water piping route of a fuel cell energy network with a concentration installing method.* International Journal of Hydrogen Energy, 31(13): 1807-1818.

Ormsbee, L., Contractor, D., 1981. *Optimization of hydraulic networks*, International Symposium on Urban Hydrology, Hydraulics, and Sediment Control, Lexington, KY, 255-261.

Palmer, C.C., 1994. *An approach to a problem in network design using genetic algorithms*. PhD thesis, Polytechnic University.

Perez, R., Ineichen, P., Seals, R., Michalsky, J., Stewart, R., 1990. *Modeling daylight availability and irradiance components from direct and global irradiance*. Solar Energy 44(5):271-289.

Perez, R., Ineichen, P., Maxwell, E., Seals, R., Zelenka, A., 1992. *Dynamic global-to-direct irradiance conversion models.* ASHRAE Transactions Research Series, 354-369.

Phetteplace, G., 1995. *Optimal Design of Piping Systems for District Heating*, CRREL Report 95-17, US Army Corps of Engineers, Cold Regions Research & Engineering Laboratory.

Premkumar, G., Chu, C.H., 1999. *Telecommunications Network Design Decision: A Genetic Algorithm Approach*. eBusiness Research Center Working Paper 08-1999.

Prüfer, H., 1918. *Neuer Beweis eines Satzes über Permutationen.* Archiv für Mathematik und Physik, 27:742-744.

Raidl, G.R., Julstrom, B.A., 2003. *Edge Sets: An Effective Evolutionary Coding of Spanning Trees*. IEEE Transactions on Evolutionary Computation, 7(3):225-239.

Raidl, G.R., Koller, G., Julstrom, B.A., 2006. *Biased Mutation Operators for Subgraph-Selection Problems*. IEEE Transactions on Evolutionary Computation, 10(2):145-156.

Rohlf, F.J., 1978. *A probabilistic minimum spanning tree algrorithm*. Information Processing Letters, 7(1):44-48.

Rothlauf, F., 2002. *Representations for Genetic and Evolutionary Algorithms*. Physica-Verlay.

Rothlauf, F., 2006. *Representations for Genetic and Evolutionary Algorithms*. 2nd edition, Physica-Verlay.

Rothlauf, F., Goldberg, D.E., Heinzl, A., 2002. *Network Random Keys – A Tree Representation Scheme for Genetic and Evolutionary Algorithms*, Evolutionary Computation 10(1): 75-97.

Sakawa, M., Kato, K., Ushiro, S., Inaoka, M., 2003. *An Interactive Fuzzy Satisficing Method for Multiobjective Operation Planning in District Heating and Cooling Plants through Genetic Algorithms for Nonlinear 0-1 Programming*, Fuzzy Sets Based Heuristics for Optimization, Springer, 235-249.

Savic, D.A., Walters, G.A., 1995. *An evolution program for pressure regulation in water distribution networks*. Engineering Optimization, 24:197-219.

Savic, D.A., Walters, G.A., 1997. *Genetic algorithms for least-cost design of water distribution networks*, ASCE Journal of Water Resources Planning and Management, 123(2): 67-77.

Sayoud, H., Takahashi, K., Vaillant, B., 2001. *A Genetic Local Tuning Algorithm for a Class of Combinatorial Networks Design Problems*. IEEE Communications Letters, 5(7):322-324.

Sinclair, M.C., 1995. *Minimum cost topology optimisation of the COST 239 European*

optical network. Proceedings of the 1995 International Conference on Artificial Neural Nets and Genetic Algorithms, 26-29.

Smith D.K., Walters G.A., 2000. *An evolutionary approach for finding optimal trees in undirected networks*. European Journal of Operational Research, 120:593-602.

Söderman, J., Pettersson, F., 2006. *Structural and operational optimisation of distributed energy systems,* Applied Thermal Engineering, 26:1400-1408.

Tang, K.S., Man, K.F., Ko, K.T., 1997. *Wireless LAN design using hierarchical genetic algorithm.* Proceedings of the Seventh International Conference on Genetic Algorithms, 629-635.

TDD, 2000. *Working Paper for Centralized District Water-cooled Air-conditioning System*, Territory Development Department, Hong Kong SAR Government.

Thevenard, D., Brunger, A., 2001. *ASHRAE research project 1015-RP: typical weather years for international locations, final report.* American Society of Heating, Refrigerating, and Air-conditioning Engineers, Inc., Atlanta.

Thiel, J., Voss, S., 1994. *Some experiences on solving multiconstraint zero one knapsack problems with genetic algorithms*. INFOR J. 32:226-242.

Thompson, E., Paulden, T., Smith, D.K., 2007. *The Dandelion Code: A New Coding of Spanning Trees for Genetic Algorithms*. IEEE Transactions on Evolutionary Computation, 11(1):91-100.

Vadrot, A., Delbès, J., 1999. *District Cooling Handbook*, European Marketing Group District Heating and Cooling.

Walters, G.A., Smith, D.K., 1995. *Evolutionary Design Algorithm for Optimal Layout of Tree Networks.* Engineering Optimization, 24:261-281.

Walters, G.A., Halhal D., Savic, D., Ouazar, D., 1999. *Improved design of "Anytown" distribution network using structured messy genetic algorithms.* Urban Water, 1:23-38.

Will, H.M., 1999. *The First Century of Air Conditioning*, American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.

Wright, J.A., 1996. *HVAC optimisation studies: Sizing by genetic algorithm*. Building Serv. Eng. Res. Technol. 17(1): 7-14.

Yamada, T., Nakano, R., 1996. *Scheduling by Genetic Local Search with Multi-Step Crossover*. Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, Berlin, Germany, September 22-26:960-969.

Zhang, Z., 1999. *Fluid Transients and Pipeline Optimization Using Genetic Algorithms*. MSc Thesis, University of Toronto.

Zhou, G., Gen, M., 1997. *A Note on Genetic Algorithms for Degree-Constrained Spanning Tree Problems,* Networks, 30(2):91-95.

Zhou, G., Gen, M., 1998. *An effective genetic algorithm approach to the quadratic minimum spanning tree problem*. Computers Ops Res, 25(3):229-237.

Appendix I

List of Publications

1. Apple L.S. Chan, Vic I. Hanby and T.T. Chow, **Optimization of distribution piping network in a district cooling system using genetic algorithm with local search**, Energy Conversion and Management. 2007, 48(10), pp. 2622-2629.

2. Apple L.S. Chan, Vic I. Hanby and T.T. Chow, **Application of Genetic Algorithm with Local Search in Optimal Piping Network Design of a District Cooling System**, Proceedings of the 6th International Conference on Indoor Air Quality, Ventilation & Energy Conservation in Buildings, Sendai, Japan, 28-31 October, 2007.

3.*** Apple LS Chan, TT Chow, Square KF Fong, John Z Lin, **Generation of a typical meteorological year for Hong Kong**, Energy Conversion & Management, 47(2006), pp.87-96.

4. Apple LS Chan, TT Chow, Square KF Fong, John Z Lin, **Performance evaluation of district cooling plant with ice storage**, Energy, 31 (2006), pp.2414-2426

*** *The Typical Meteorological Year (TMY) developed in this PhD study had been accepted by the U S Department of Energy and is now available in EnergyPlus weather format from the following web site.*

http://www.eere.energy.gov/buildings/energyplus/cfm/weather_data3.cfm/region=2_asia_wmo_region_2/country=CHN/cname=China

Appendix II

List of Details and Examples of Various Encoding Methods

This Appendix gives detailed steps of encoding and decoding of tree network using various encoding methods including Prüfer number, characteristic vector, link and node biased encoding, network random keys and direct tree encoding. Illustrative examples (Rothlauf, 2002) are also presented for easy reference.

II-1. Prüfer Number

The Prüfer number encodes an n-node tree with a string of length $n - 2$ and each element of the string is of base n. As the mapping is one-to-one, a Prüfer number is a unique encoding of a tree, and there are n^{n-2} different possible Prüfer numbers.

Below is a summary of constructing Prüfer numbers for a tree configuration.

(i) Let i be the lowest numbered node of degree 1 in the tree.
(ii) Let j be the one node which is connected to i (there is exactly one). The number of the jth node is the furthest right digit of the Prüfer number.
(iii) Remove node i and the link (i, j) from the tree and from further consideration.
(iv) Go to (i) until only two nodes (that means one link) are left.

The construction of a tree from the Prüfer number follows the construction of the Prüfer number from a tree. It goes as follows:

(i) Let P be a Prüfer number with n-2 digits. All node numbers which are not in P can be used for the construction of the tree. (n-1 is the number of links in a tree structure.)
(ii) Let i be the lowest numbered eligible node. Let j be the leftmost digit of P.
(iii) Add the link (i, j) to the tree.
(iv) Designate i as no longer eligible and remove the leftmost digit j from the Prüfer number.
(v) If j does not occur anywhere else in the remaining Prüfer number, designate j as eligible.

(vi) Go to (ii) until no digits remain in the Prüfer number. If no digits are left, then there are exactly two number, r and s, which are eligible. Finally, add the link (r, s) to the tree.

In Figure II-1 shown below, there are 6 nodes in the network. Therefore there should be 4 digits in the Prüfer number. The lowest numbered node with degree 1 is node 1. This node is connected to node 2 so the Prüfer number starts with a 2. Then node 1 is removed from further consideration and the lowest numbered node with degree 1 is to be searched. Node 2 is identified which is connected to node 5. The Prüfer number becomes 25. After removing node 2, node 3 is the lowest numbered node which is eligible. Node 3 is connected to 6 so that the Prüfer number becomes 256. The node 4 is the lowest eligible node and then 5 is added to the Prüfer number. Finally, only two nodes remain in the network. The procedure stops and the resulting Prüfer number is 2565.

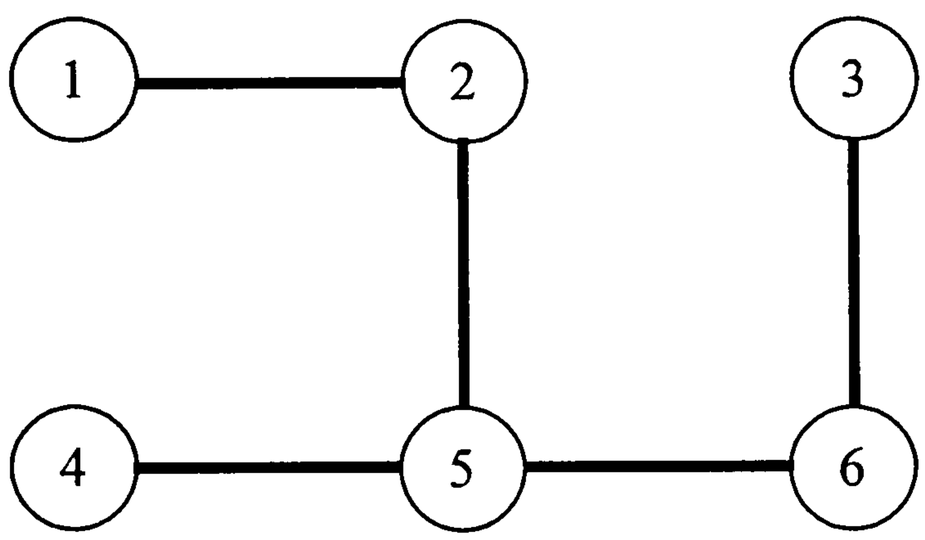


Figure II-1 A tree and the corresponding Prüfer number $P = 2565$

For constructing the network from the Prüfer number $P = 2565$. Eligible nodes are 1, 3 and 4. As 1 is the lowest eligible node, and 2 is the leftmost digit of the Prüfer number, link (1, 2) is added to the network. 1 is then no longer eligible, and 2 does not occur anywhere else in the string. Therefore, the nodes 2, 3 and 4 are eligible and P becomes 565. Now 2 is the lowest eligible node and link (2, 5) is added to the network. As 5 occurs somewhere else in the string, 5 is not designated as eligible. Thus, only 2 is removed from the pool of eligible numbers. And then a link (3, 6) is added to the network. Now, only the nodes 4 and 6 are eligible and $P = 5$. Then link (4, 5) is added. Finally, all the digits are removed from P and the numbers 5 and 6 remain eligible. The link (5, 6) is added to the network and the procedure terminates. The network is constructed as shown in Figure II-1.

II-2. Characteristic Vector

A characteristic vector is a binary vector that indicates if a link is used or not in a network. For an n-node network, there exist $n(n-1)/2$ possible links, and a characteristic vector of length $l = n(n-1)/2$ is necessary for encoding the structure of an n-node network. All the possible links must be numbered and each link must be assigned to a position in the vector. In Table II-1, an example of a characteristic vector for a 5-node tree is given. The nodes are labelled from A to E. The link from node A to B is assigned to the first position in the string, the link from A to C is assigned to the second position, and so on. To indicate if the i^{th} link is established, the value at position i is set to one. If no link is established, the value of the allele is set to zero. The tree represented by Table II-1 is shown in Figure II-2.

Table II-1

The characteristic vector for the tree shown in Figure II-2

0	1	0	0	0	1	0	1	0	1
A - B	A - C	A - D	A - E	B - C	B - D	B - E	C - D	C - E	D - E

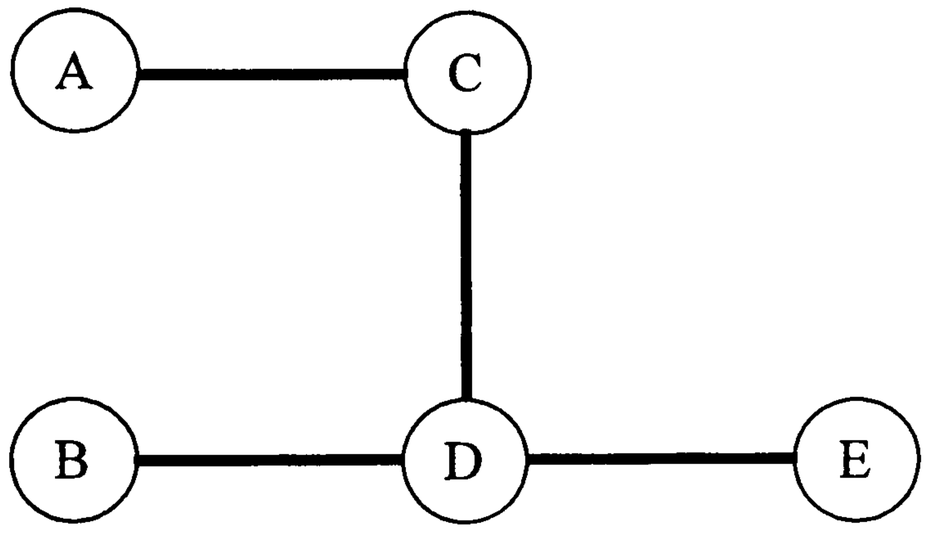


Figure II-2 A five node tree

II-3. Link and Node Biased Encoding

In node-biased encoding, the chromosome $\boldsymbol{b}$ holds the biases for each node, and has length n for an n node network. The values in the distance matrix d_{ij} are modified according to $\boldsymbol{b}$ using the weight function in Equation (II-1).

$$d'_{ij} = d_{ij} + p\left(b_i + b_j\right)d_{\max} \tag{II-1}$$

The bias b_i is a floating number between zero and one, d_{max} is the largest value in the distance matrix and p controls the influence of the biases.

For example, a vector $b = \{0.7, 0.5, 0.2, 0.8, 0.1\}$ is a node-biased vector and holds the bias for each node. A distance matrix for the 5-node problem is defined as:

$$D = \begin{pmatrix} - & 2 & 1 & 3 & 4 \\ 2 & - & 5 & 6 & 3 \\ 1 & 5 & - & 4 & 3 \\ 3 & 6 & 4 & - & 10 \\ 4 & 3 & 3 & 10 & - \end{pmatrix}$$

For the construction process, all the values of the modified distance matrix have to be calculated. Using $p = 1$, $d'_{0,1} = 2 + (0.7 + 0.5) * 10 = 14$. All the modified distances $d'_{i,j}$ are calculated in the similar way according to Equation (II-1), the modified distance matrix will be as follows:

$$D' = \begin{pmatrix} - & 14 & 10 & 18 & 12 \\ 14 & - & 12 & 19 & 11 \\ 10 & 12 & - & 14 & 6 \\ 18 & 19 & 14 & - & 19 \\ 12 & 11 & 6 & 19 & - \end{pmatrix}$$

Using Prim's algorithm for the modified distance matrix D', the tree finally obtained is illustrated in Figure II-3. The represented tree is calculated as the minimum spanning tree using the distance matrix D'. For example, $d'_{0,2} = 10 < d'_{0,i}$, where $i \in \{1, 3, 4\}$. Because the link between node 0 and node 2 has the shortest distance d', it is used for the represented tree.

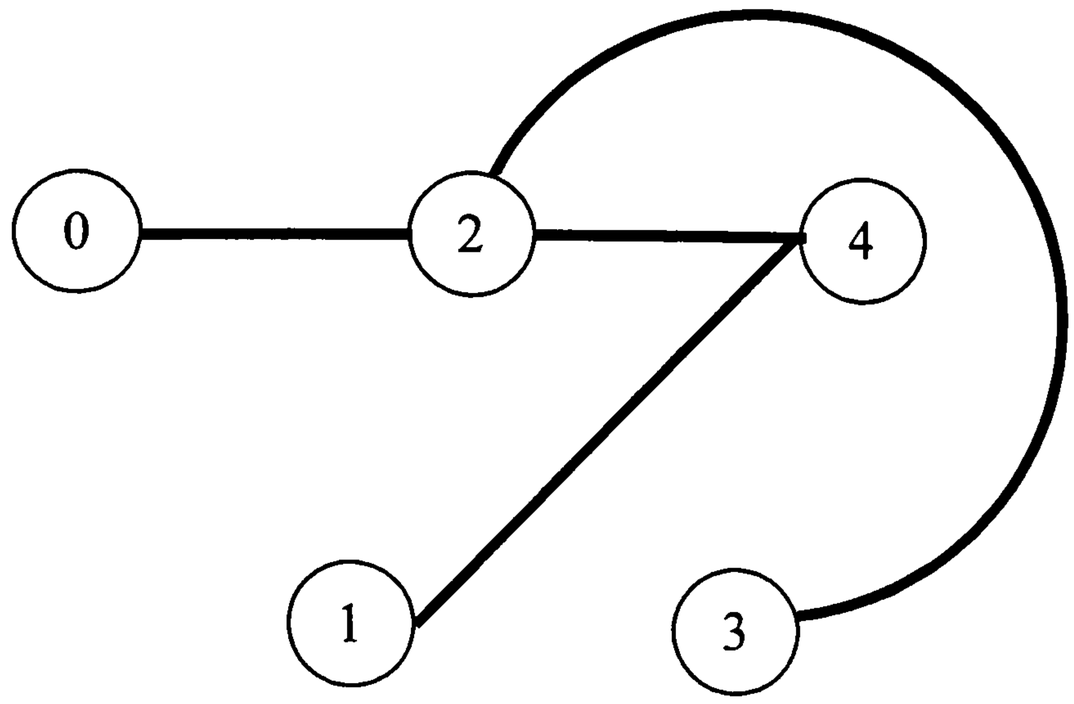


Figure II-3 An example tree for the node-biased encoding

However, not all the possible tree candidates can be encoded by the node-biased encoding. In order to overcome this problem, an extended version of the encoding with an additional link-bias is introduced. The chromosome holds biases not only for the n nodes but also for all possible $n(n-1)/2$ links, and has overall length $l = n(n-1)/2$. The weighting function for the elements in the distance matrix is extended to:

$$d'_{ij} = d_{ij} + P_1 b_{ij} d_{\max} + P_2 \left(b_i + b_j\right) d_{\max} \qquad \text{(II-2)}$$

with the link-specific bias $b_{i,j}$, the weight of the link-specific bias P_1, and the weight of the node-specific bias P_2. Using this encoding, it can represent all possible trees. The following example illustrates the link-and-node-biased encoding. This example chromosome holds the node bias {0.7, 0.5, 0.2, 0.8, 0.1} and the link bias:

$$\begin{pmatrix} - & 0.1 & 0.6 & 0.2 & 0.8 \\ 0.1 & - & 0.1 & 0.9 & 0.5 \\ 0.6 & 0.1 & - & 0.3 & 0.2 \\ 0.2 & 0.9 & 0.3 & - & 0.4 \\ 0.8 & 0.5 & 0.2 & 0.4 & - \end{pmatrix}$$

With $P_1 = 1$ and $P_2 = 1$ and using the distance matrix from Equation (II-2), for example,

$d'_{0,1} = 2 + 0.1*10 + (0.7 + 0.5)*10 = 15$. Consequently, the modified distance matrix is obtained as:

$$D' = \begin{pmatrix} - & 15 & 16 & 20 & 20 \\ 15 & - & 13 & 28 & 16 \\ 16 & 13 & - & 17 & 8 \\ 20 & 28 & 17 & - & 23 \\ 20 & 16 & 8 & 23 & - \end{pmatrix}$$

With this modified distance matrix D', the tree can be obtained as shown in Figure II-4 below:

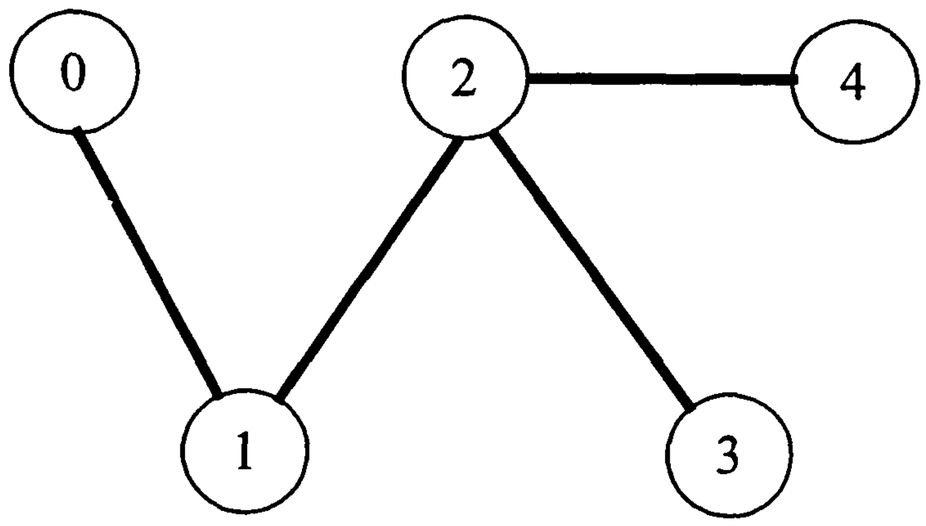


Figure II-4 An example tree for the link-and-node-biased encoding

II-4. Network Random Keys

When constructing the tree, the positions of the keys in the key sequence *r* are interpreted in the same way as for the characteristic vector. The positions are labelled and each position represents one possible link in the tree. From a key sequence *r* of length $l = n(n-1)/2$, a permutation r^s of l numbers can be constructed. Then the tree is constructed from the permutation r^s as follows:

(i) Let $i = 0$, G be an empty graph with n nodes, and r^s the permutation of length $l = n(n-1)/2$ that can be constructed from the key sequence r. All the possible links of G are numbered from 1 to l.

(ii) Let j be the number at the ith position of the permutation r^s.

(iii) If the insertion of the link with number j in G would not create a cycle, then insert the link with number j in G.

(iv) Stop, if there are $n - 1$ links in G.

(v) Increment i and continue with step (ii).

With this calculation rule, a unique and valid tree can be constructed. The functionality of the NetKey encoding can be illustrated by the following example. A key sequence is listed in Table II-2. The Permutation $r^s = 10 \rightarrow 8 \rightarrow 6 \rightarrow 9 \rightarrow 2 \rightarrow 7 \rightarrow 1 \rightarrow 5 \rightarrow 4 \rightarrow 3$ can be constructed from the random key sequence r. Starting to construct the graph G by adding the link D – E (position 10) to the tree. This is followed by adding C – D (position 8) and B – D (position 6). If the link C – E (position 9) is added to the graph, the cycle C – E – D – C would be formed, so C – E is skipped and A – C (position 2) is continuously added. Then a tree with four edges is formed (shown in Figure II-5) and the construction algorithm is terminated.

Table II-2

The key sequence for an example with a five node tree

Position	1	2	3	4	5	6	7	8	9	10
Value	0.55	0.73	0.09	0.23	0.40	0.82	0.65	0.85	0.75	0.90
Link	A - B	A - C	A - D	A - E	B - C	B - D	B - E	C - D	C - E	D - E

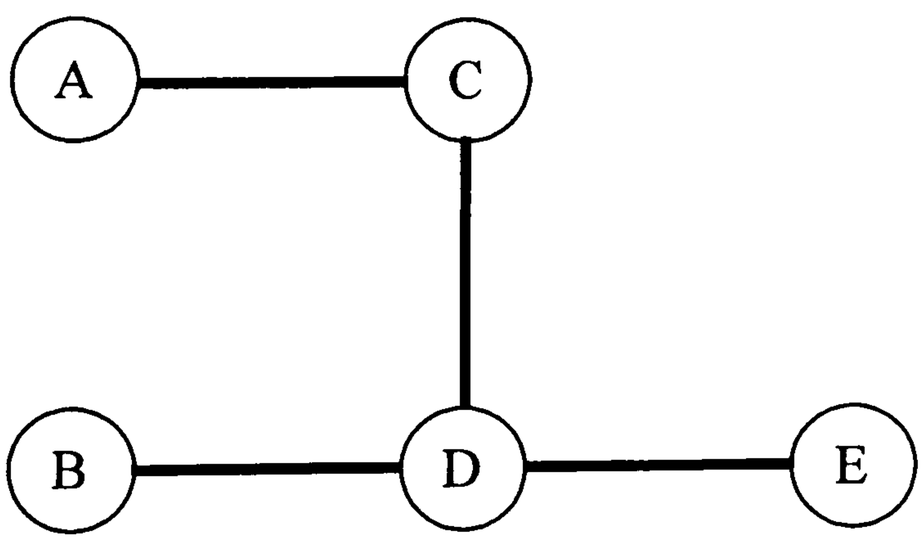


Figure II-5 A five node tree

II-5. Direct Tree Encoding

In direct tree encoding, there is no specific genotype(Φ_g)-phenotype(Φ_p) mapping $f_g : \Phi_g \rightarrow \Phi_p$ anymore but work directly on the phenotype. In contrast to the indirect encoding, where the genotypic space is different from the phenotypic space, $\Phi_g \neq \Phi_p$, the mutation and crossover operators are directly applied to the phenotypes $x_p \in \Phi_p$.

The situation is illustrated in Figure II-6. In direct encoding, the structure of the problem is directly represented and there is no need to design a proper and efficient encoding. However, since there are no standard mutation and crossover operators that can be used, it is difficult to design proper problem-specific operators for the problems with direct encoding.

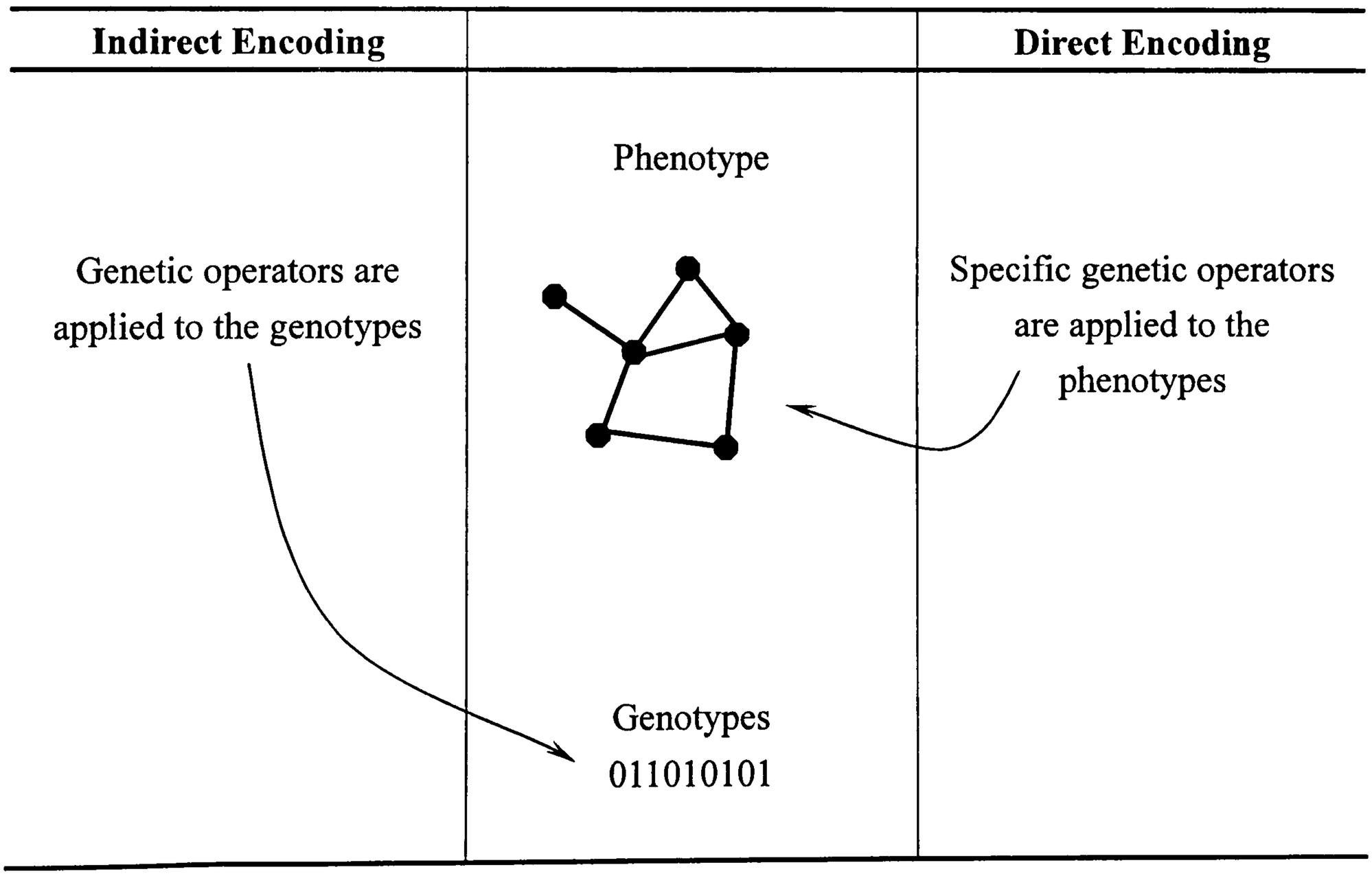


Figure II-6 Difference between direct and indirect encoding